
TEKNILLINEN

TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto
Informaatiotekniikan laitos

MIKKO KIVIHARJU

MONIKANAVAISIA PALVELUPYYNTÖJÄ KÄSITTELEVÄN UNIVERSAL SERVICE QUEUE – MALLIN SUUNNITTELU JA PILOTOINTI

Diplomityö, joka on opinnäytteenä jätetty tarkastettavaksi diplomi-insinöörin tutkintoa varten.

Espoossa 20.01.1999

Työn valvoja: Professori Olli Simula

Työn ohjaaja: DI Tapio Huomo

TEKNILLINEN KORKEAKOULU **DIPLOMITYÖN TIIVISTELMÄ**

Tekijä: Mikko Kiviharju

Työn nimi: Monikanavaisia palvelupyyntöjä käsittelevän Universal Service Queue - mallin suunnittelu ja pilotointi

Päivämäärä: 20.1.1999 **Sivumäärä: 115**

Osasto: Tietotekniikka

Professuuri: Tik-115 Informaatiotekniikka

Työn valvoja: Professori Olli Simula

Työn ohjaaja: DI Tapio Huomo

Tämän työn tavoitteena oli mediariippumattoman, monikanavaisia palvelupyyntöjä käsittelevän järjestelmän mallin (USQ) suunnittelu ja pilotointi. Tavoite pyrittiin saavuttamaan tutkimalla alueeseen liittyviä standardeja, konstruoimalla näiden pohjalta USQ-malli, sekä pilotoimalla (mallin testaus) tätä PSTN / IP - medioiden osalta.

Työ on jaettu neljään osaan. Ensimmäisessä osassa tarkastellaan palvelupyyntöjen käsittelyteknologian alustajärjestelmiä ja niihin liittyviä käsitteitä. Tässä osassa kuvataan palvelupyyntöjen ohjausjärjestelmää sellaisena kuin se oli ennen monikanavaisuuden käsitettä. Samalla selvitetään alueeseen liittyviä esitietovaatimuksina pidettäviä käsitteitä.

Työn toinen osa muodostuu kirjallisuustutkimuksesta, jossa tutkitaan USQ:n mallintamiseen mahdollisesti tarvittavia standardeja ja rajapintoja. Toisessa osassa lukija perehdytetään syvällisemmin itse määrittelyosiossa vaadittuihin käsitteisiin ja käyttöönotettujen ratkaisujen syihin ja seuraamuksiin. Tässä osassa osoitetaan myös suuntaviivoja jatkokehitystä varten.

Kolmas osa muodostuu itse USQ-mallin määrittelystä ja sen toteutusmahdollisuuksista. Määrittelyosiossa syvennetään USQ:n käsitettä ja osoitetaan yleisiä konstruointikeinoja USQ:ta tukeville järjestelmille. Toteutusmahdollisuuksia pohtivassa osiossa on esitetty useita eri skenaarioita mahdollisille implementoinneille ottaen huomioon olemassaoleva tekniikka. Esitettyjen skenaarioiden yleistystä peilataan vielä lopuksi alan standardeja vasten auttamaan neljännen osan toteutuksen arvioinnissa.

Neljäs ja viimeinen osa sisältää kuvauksen pilottina tehdystä ohjelmistosta arkkitehtuurikuvauksineen ja teknisine selvityksineen. Tässä osassa esitetty malli testataan toimivuutensa ja toteutuskelpoisuutensa puolesta ja esitetään joitakin ideoita jatkokehityksen osalta. Yksityiskohtaisemmat tekniset dokumentit ilmenevät liitteistä, joissa on myös taulukoitu ohjelmiston käyttämät viestit verrattuna standardeissa esitettyihin viesteihin.

Alkulause

Tämä diplomityö on tehty HM&V Telecommunications Oy:ssä liittyen CustomerConnect-puhelunohjausohjelmiston jatkokehitystyöhön. Työn valvojana toimi informaatiotekniikan laitoksen professori Olli Simula, jota haluan kiittää hänen osuudestaan. HM&V Telecommunications Oy:n puolesta ohjaajanani toiminutta konsultti Tapio Huomoa kiitän hänen arvokkaista ideoistaan koskien erityisesti työn teoriapainottaista osuutta.

Kiitokset kuuluvat koko HM&V Research Oy:n ja HM&V Telecommunications Oy:n henkilökunnille asiantuntemuksesta ja hyvästä työilmapiiristä. Erityisesti haluan kiittää ohjelmistotiimin jäseniä Jyrki Korhosta ja Arthur van der Knaapia heidän ohjelmointiosuudessa tarjonneestaan korvaamattomasta avusta. Kunnia alkuperäisestä Universal Service Queuen ideasta kuuluu toimitusjohtaja Matti Mäkelinille, ja kiitokset oikaisuista haluan esittää varsinkin Hannu Tuomisaarelle hänen standarditutkimusta koskeneista kommenteistaan.

Tämä työ on osa CustomerConnect-ohjelmistoa ja sen dokumentointia, joten siihen ovat välillisesti antaneet osansa kaikki CustomerConnect-projektissa mukana olleet henkilöt, joista jokaista haluan tässä kiittää.

Lämpimät kiitokset haluan lausua myös vanhemmilleni sekä korvaamattomasta avusta työn lyhennelmän kääntämisessä että arvokkaasta henkisestä tuesta koko tämän pitkän prosessin aikana.

Lopuksi osoitan vielä sydämelliset kiitokseni Niinalle, jonka tarjoama henkinen tuki oli korvaamatonta, ja joka jaksoi uskoa tämän työn valmistumiseen jatkuvista viivytyksistä huolimattakin.

Espoossa, 20. Tammikuuta 1999

Mikko Kiviharju

Lyhenteet

ACD	Automatic Call Distribution
ACL	A Controlling Language
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
AV	Audio-Video
AXI	ActiveX Interface (CustomerConnect-IPCC-komponentin yhteydessä)
C	Computing (-domain, CSTA:n yhteydessä)
CC	Customer Connect
CGI	Common Gateway Interface
CID	Call ID
COM	Component Object Model
CPU	Central Processing Unit
CRV	Call Reference Value
CSTA	Computer Supported Telephony Applications
CT	Computer Telephony
CTI	Computer Telephony Integration
DCOM	Distributed Component Object Model
DSP	Digital Signal Processor
DTMF	Dial Tone Modulation Format
ECMA	European Computer Manufacturers Association
FIFO	First In - First Out
FTP	File Transfer Protocol
GK	Gatekeeper
gln	global logical number (CC:n hajautettavuusluvun yhteydessä)
GRQ	Gatekeeper Request
GUI	Graphical User Interface
GW	Gateway
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IADM	InterServer Administration Interface (CustomerConnectin MCU-toteutuksen yhteydessä)
IAPI	InterServer API (CustomerConnectin MCU-toteutuksen yhteydessä)
iCAPI	IP Customer Client API (CustomerConnectin MCU-toteutuksen yhteydessä)
IIS	Internet Information Server
ILS	Internet Locator Service
IN	Intelligent Network
IP	Internet Protocol
IPCC	IP Customer Client (CustomerConnectin yhteydessä)
IPX	IP X-server (IP vaihdeserveri CustomerConnectin yhteydessä)
IRC	Internet Relay Chat
ISAPI	Internet Server API
ISDN	Integrated Services Digital Network
ITU-T	International Telecommunication Union - Telecommunication standardization section
IUR	Interactive USQ Response

IVR	Interactive Voice Response
IWR	Interactive Web Response
iXAPI	InterServer API (CustomerConnectin MCU-toteutuksen yhteydessä)
JDK	Java Development Kit
LAN	Local Area Network
lln	local logical number (CC:n hajautettavuusluvun yhteydessä)
MAPI	Mail API
MC	Multipoint Controller
MCU	Multipoint Control Unit
MP	Multipoint Processor
MSP	Media Service Provider
MSVJ	Microsoft Visual Java
NM	NetMeeting
NSAPI	Netscape Server API
NT	New Technology
ODBC	Open DataBase Connectivity
OLE	Object Linking and Embedding
OSI	Open Systems Interconnection
PBX	Private Branch eXchange
PC	Personal Computer
PIN	Personal Identification Number
POTS	Plain Old Telephony System
PSTN	Public Switched Telephone Network
PTP	Point-To-Point (ILS:in yhteydessä)
QoS	Quality of Service
RAM	Random Access Memory
RAS	Registration, Admissions and Status
RDBMS	Relational DataBase Management System
RFC	Request For Comments
RM	RealMedia komponentti (arkkitehtuurikuvauksissa)
RPN	Response Priority Number (USQ-teorian yhteydessä)
RSVP	Resource Reservation Protocol
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
S	Switching (-domain, CSTA:n yhteydessä)
SCN	Switched Circuit Network
SCS	Simple Call State
SCTP	Simple Computer Telephony Protocol
SDK	Software Development Kit
SMTP	Simple Mail Transfer Protocol
SR	Special Resource (-domain, CSTA:n yhteydessä)
SS	State Server (CustomerConnect:in yhyteydessä)
SWID	Switch ID
T	Terminal (H.323:n arkkitehtuurikuvauksissa)
TAPI	Telephony Application Programming Interface
TCP/IP	Transport Control Protocol / Internet Protocol
TID	Transaction ID
TLB	Type Library
TSP	Telephony Service Provider

UI	User Interface
ULS	Universal Locator Service (ILS:n edeltäjä)
URL	Uniform Resource Locator
USQ	Universal Service Queue
VM	Virtual Machine (Java-kielen yhteydessä)
VRU	Voice Response Unit
WWW	World Wide Web

Keskeiset termit

CSTA	Computer Supported Telephony Applications, ECMA-standardi (European Computer Manufacturers Association), joka määrittelee tiettyjä puhelu- ja vaihdepalveluja erit. call center tyyppiseen ympäristöön.
CTI	Computer Telephony Integration, tietokoneen ja telekommunikaation yhdistämiseksi tarkoitetut teknologiat ja tekniikat
H.323	ITU-T:n julkaisema suositus multimediayhteydenpitoon alunperin lähiverkon (LAN) ylitse. Sisältää koodekit, arkkitehtuuri- ja ohjausprotokollasuositukset
IN	Intelligent Network, älyverkko, julkisten puhelinverkkojen "älykkäitä solmuja" sisältävä osa.
Inbound puhelu	Puhelu, joka tulee tarkasteltavan järjestelmän ulkopuolelta itse järjestelmään sisään, esim. palvelupyyntö.
IP-puhelu	IP-protokollalla toimivien verkkojen l. Internetin yli tapahtuvia reaaliaikaisia audio-, video- ja datayhteydenpitoja tai niiden joitakin osajoukkoja kahden tai useamman osapuolen välillä
IVR	Interactive Voice Response, järjestelmä, joka pystyy DTMF-puhelimilla tai tietojärjestelmillä ohjatusti soittamaan nauhoitetta tms. soittajalle. Järjestelmä sisältää valikko-ohjauksen muillekin toiminnoille.
IWR	Interactive Web Response, periaatteessa sama kuin IVR, mutta tarkoitettu selaimen yli käytettäväksi. Tähän mennessä ilmestyneet IWR-systeemit näyttävät katsojalle videostreamien lisäksi myös nk. mikrosaitteja, l. IWR-järjestelmän valvonnan alla olevia sivuja.
Java	Erityisesti HTML-dokumenteissa käytetty välikoditasolla tulkattu alustariippumaton suuren suosion saavuttanut ohjelmointikieli, C++:n osajoukko.
Looginen numero	Erit. PBX:ssä sellainen tyypillisesti analoginen liittymä, jossa ei ole puhelinlaitetta kiinni. Käytetään IP-puheluissa esittämään tiettyyn ryhmään assosioitavaa IP-

	osoitetta, joka ei vastaa mitään laillista IP-osoitetta.
Media	Yleisnimi eri tavoin tulkittaville informaation esitystavoille, esim. audio, video ja data. Tässä dokumentissa medialla ja mediatyypillä erotetaan myös eri siirtoteitä kulkevat audiotyypit, kuten IP- ja PSTN-verkoissa kulkeva audio
Middleware	Useampikerrosarkkitehtuurissa keskelle kytkettävä ohjelmisto automaattisesti ohjattavine rajapintoineen.
Multicast conferencing	Erotuksena Ad Hoc-konferenssille: konferenssi, joka säästää kaistanleveyttä ohjaamalla mediavirrat keskitetysti heti, kun mahdollista (käytännössä silloin, kun mediavirta siirtyy LANin ulkopuolelle)
POTS	Plain Old Telephony System, epävirallinen ilmaus, jota käytetään ilmaisemaan perinteistä puhelumaailmaa ja peruspäätelaitteita.
PSTN	Public Switched Telephone Network = konventionaalinen puheluverkko, televerkko. Myös nimellä GSTN, General Switched Telephone Network.
Outbound – puhelu	Puhelu, joka tehdään tarkasteltavan järjestelmän piiristä ulospäin.
RAS	Registration, Admissions and Status, H.323:n määrittelemä viestijoukko terminaalin kommunikoimiseksi gatekeeperin kanssa.
RFC	Request For Comments, kenen tahansa Internetiin postaama ehdotus standardiksi. Numeroidaan 1:stä eteenpäin, tämän kirjoitushetkellä menossa n. 1400:ssa.
RTP	Real-time Transport Protocol, IETF:n standardiprotokolla, joka on suunniteltu käytettäväksi multimediastreamien siirtelyyn Internetin yli.
SCN	Switched-Circuit-Network, esim. normaali PSTN, missä yhteydelle on reitin jokaisessa noodissa varattu konkreettisia piirejä (circuits)
SCTP	Simple Computer Telephony Protocol, verraten uusi, mahdollisimman alustariippumaton tekstipohjainen

	protokolla CT-sovellusten viestinvälitykseen.
Screen based telephony	Jonkin puhelinlaitteen (tyypillisesti alaliittymän) ohjaaminen puhelinlaitteen yhteydessä olevalta näyttöpäätteeltä.
VRU	Voice Response Unit, yksinkertainen IVR, pelkkä DSP-kortti (Digital Signal Processing), joka pystyy soittamaan ääntä (tyyp. musiikkia) siinä oleville linjoille.

1. Johdanto

1.1 Taustaa

Tietokoneiden verkottuessa ja varustetason parantuessa niiden käyttötarkoitus on olennaisesti laajentunut. Alunperin laskentaan ja sittemmin tekstinkäsittelyyn käytetty tietokone on saanut uusia tehtäviä ihmisten välisen viestinnän välityssolmuna ja jopa itse viestintävälineenä. Audio- ja videovirta ei kulje enää pelkästään puhelimen ja tarkoitukseen suunniteltujen erikoislaitteistojen kautta. Tietokoneita ei käytetä enää vain tekstimuotoisten viestien välitykseen, vaan se on alkanut näytellä hyvin aktiivista roolia itse mediavirtojen päätepisteenä [HANN96]. Internetin anarkistinen luonne antaa tietokoneelle jopa erinäisiä etuja perinteisiin AV-virtaa käsitteleville systeemeille: mediavirran siirtotie ei ole kokonaan kenenkään hallinnassa, mikä laskee yhteyskustannuksia perinteisiä systeemejä alemmalle tasolle; siirtotien erottaminen muusta verkkoaktiviteetista tekee tietokonetta viestimenään käyttävän henkilön käytännössä riippumattomaksi fyysisestä sijainnista ja kommunikaation kytkeminen muihin informaatiojärjestelmiin on huomattavasti helpompaa perinteisiin systeemeihin verrattuna. Lisäksi Internet tietokoneiden välisenä verkkona on länsimaissa nykyisin laajalle levinnyt ja kasvuvauhdin ollessa eksponentiaalinen alkaa viestipäätteinä käytettävien tietokoneiden tiheys olla jo vertailtavissa tavallisten viestimien tiheyden kanssa (n. 70 miljoonaa Internetiin liitettyä tietokonetta on n. kymmenen prosenttia puhelinverkossa olevien tavallisten viestimien liittymien määrästä; on ennustettu [HANN96], että vuoden 2000 jälkeen internetiin liittyneitä laitteita on jo saman verran kuin puhelinlaitteita.). Edellämainituista syistä johtuen tietokoneet mediavirran tuottajina ovat varteenotettava vaihtoehto perinteisten mediavirran lähteiden kuten puhelimien ja neuvottelujärjestelmien rinnalle.

Puhdas tietokonepohjainen multimediaviestintä Internetissä on kehittynyt kuten alustansakin: suhteellisen anarkistisesti. Multimediaviestintää kehittämässä on ollut lukuisia eri valmistajia ja palvelun tuottajia, joista osa on suuria ohjelmisto- tai telealan jättejä, ja osa pieniä muutaman hengen autotalliyrityksiä. Standardointiyrityksiä multimediaviestinnälle on ilmennyt jo koko multimedia-käsitteen syntymästä alkaen. Valitettavasti ensimmäisten standardointiyrityksien aikainen laitteisto ei sallinut kovin kattavia verkostoja toteuttamaan näitä standardeja. Tuloksena oli laaja valikoima järjestelmiä, jotka eivät suostuneet ollenkaan kommunikoimaan keskenään [HUOM93, DAWS96c]. Vasta nyt on uuden standardin mahdollista saavuttaa tarpeeksi laaja suosio ollakseen muutakin kuin pelkkä kuriositeetti. Tähän vaikuttavat erityisesti Internetiin liittyneiden päätteiden suuri lukumäärä, sekä allaolevien siirtoteiden nopeus: jopa videon siirto on mahdollista minkä tahansa Internetiin liittyneiden päätteiden välillä.

Tietokonepohjaisen multimediaviestinnän kehityksen tavasta voidaan päätellä eräs ilmiselvä tosiseikka: ensimmäisessä vaiheessa tärkeintä on yhteyksien rakentaminen sekä teknologiaa että ohjelmistoja kehittämällä. Tästä on seurauksena teknologialtaan melko edustava, mutta erittäin kontrolloimaton usean pisteestä pisteeseen kulkevan mediavirran verkosto [HANN96]. Verkoston hallintaa ei varsinaisesti ole olemassa, eikä siihen voi edes luoda keskitetysti hallittuja komponentteja tai aliverkkoja olemassaolevin sovelluksin. On tarvetta keskitetylle hallintajärjestelmälle, joka ei saa olla liian eristetty IP-maailman (Internet ja sen infrastruktuuri) ulkopuolella olevista järjestelmistä. On huomattava, että keskittäminen ei ole itsetarkoitus: pyrkimyksenä ei ole mikään globaali keskittäminen, vaan erikokoisten yksikköjen luonti nykyisten lähes täydellisen anarkistisen järjestelmien rinnalle [HANN96].

PSTN-maailmassa (PSTN = Public Switched Telephony Network) keskitetty mediavirranohjaus on pitkälle kehittynyttä. Tämä johtuu useasta seikasta, jotka loppujen lopuksi johtuvat samasta asiasta: PSTN-verkon iästä. Koska PSTN-verkko on rakennettu yli sadan vuoden aikana, se kantaa mukanaan joitakin jäänteitä menneisyydestä. Tärkeimpiin näistä jäänteistä kuuluvat kytkentämalli ja topologia. Internet on perusfilosofialtaan mahdollisimman vähän kuroumia (so. pisteitä, joiden poistaminen aiheuttaa verkon pilkkoutumisen kahdeksi tai useammaksi aliverkoksi) sisältävä silmukoitu verkko, joka on kasattu enemmän bottom-up – mallin kuin top-down-mallin mukaisesti [STAL94]. PSTN-verkko on taas hyvin keskitetty: se on tehty top-down mallilla runkoverkon ulkopuolella kasvattamalla tiettyihin kytkentämatriisikeskuksiin (älyverkkoon) lisähaaroja pienempiin vaihteisiin (PBX, Private Branch eXchange) ja yksittäisiin puhelinlaitteisiin, joskin itse runkoverkon topologiana on silmukoitu verkko [HANN96]. Syntytavasta johtuen PSTN-verkko on piirikytkentäinen, l. kommunikointireitti on syntymisensä jälkeen vakio, mikä taas ei päde Internetiin. Internet suunniteltiin alun alkaen sellaiseksi, että jonkin siirtotien katkeaminen ei vaikuta yhteyksiin niin kauan kuin päätepisteiden välillä on jokin polku. Tällöin piirikytkentäisyys ei käy, vaan vaaditaan dynaamisempi järjestelmä eli pakettikytkentäisyys.

PSTN:n, PBX:ien, älyverkkojen ja puhelinlaitteiden muodostama PSTN-maailma ja Internetin, sen sovellusten, standardien ja palvelimien muodostama IP-maailma ovat hyvin erilaisia, mutta kummallakin on tarjottavaa toisilleen: IP-puolella on hallussaan verkon solmujen sisältämien palvelimien kautta monipuolisempi mediatarjonta ja suurempi tietojenkäsittelykapasiteetti, kun taas PSTN:llä on pidemmälle viety keskitetyn ohjauksen logiikka ja tällä hetkellä myös suurempi volyymi verkon solmujen osalta.

PSTN-ja IP-maailman yhdistämiseksi tarvitaan nk. CT-tekniikoita. Näiden Computer Telephony-tekniikoiksi kutsuttujen menetelmien kypsyminen mahdollistaa teknologian ja eräiden sovelluskomponenttien puolesta PSTN- ja IP-puheluiden yhtenäisen ohjaamisen. Toisaalta on melko todennäköistä, että kehitys ei pysähdy pelkästään IP- ja PSTN-puheluihin, vaan muitakin yhteydenottotapoja syntyy yhä nopeammin. Vaikka tällä hetkellä

tarve on hyvin selkeä: IP- ja PSTN – maailmojen yhdistäminen IP-pohjaisia mediavirtoja keskitetysti ohjaamalla, se ei välttämättä ole riittävän yleinen [HANN96]. Ajan kuluessa kestävin järjestelmä abstrahoi IP- ja PSTN-kommunikoinnin erikoistapauksiksi yleisemmästä kommunikoinnin keskitetystä ohjaamisesta. Kommunikoinnin mediariippumaton ohjaus on tavoite, joka ei ole riippuvainen uusimmista mediatyypeistä (*) tai niiden muutoksista. Tällaisessa ohjausjärjestelmässä ytimenä on ohjauslogiikka, ja sen yhdyskäytävinä muuhun maailmaan eri medioita käsittelevät komponentit. Valitettavasti ohjauslogiikan irroittaminen mediaspesifisistä komponenteista ei ole mikään yksinkertainen asia, kuten ei myöskään anarkistisesti suunniteltujen mediavirran käsittelykomponenttien saattaminen keskitetyn valvonta- ja hallintalogiikan alle [HANN96].

Muita huomioonotettavia seikkoja ovat erot IP-puhelupakettien ja piirikytkentäisesti toimivien PSTN-verkkojen siirtämien puhelujen välillä, sekä audiovisuaalisen ja tekstimuotoisen datan erot: tarvitaan jonkinlaisia yhdyskäytäviä, joita on voitava hyödyntää järkevällä tavalla mediariippumattomuuden aikaansaamiseksi. Käytännössä yhdyskäytävät tekstin ja audion välilläkin ovat mahdollisia, mutteivät vielä riittävän kypsiä tässä dokumentissa tarvittuun reaaliaikaiseen käyttöön . Tämä diplomityö keskittyykin prototyypinomaisesti IP-puhelujen ja PSTN-puhelujen yhteiseen ohjaamiseen, pitäytyen kuitenkin laajemmissa kehyksissä sikäli, että kaikki näiden kahden eri mediatyypin yhdistämiseksi saavutetut tulokset olisivat mahdollisimman usein sinällään yleistettävissä muidenkin eri mediatyyppien ohjaukseen.

1.2 USQ-problematiikka

USQ-problematiikan (Universal Service Queue) keskeisenä alueena on karkeasti ottaen IP-puheluiden keskitettyjen ohjausjärjestelmien kehittymättömyys ja toisaalta ns. mediariippumattomien systeemien puute, johon liittyy kiinteästi käytössä olevien "standardien" varsin laaja kirjo [HANN96]. Ongelmakenttä on siis varsin suuri, ja tarvitsee täsmennystä. Tämä työ keskittyykin kolmeen ongelmakentän pääkohtaan:

- mediariippumattoman ohjausjärjestelmän teoriaan ja suunnitteluun
- edellisestä erikoistapauksena IP-mediankäsittelyominaisuuksien lisäämiseen olemassaolevaan puhelunohjausohjelmistoon

(*) Toisistaan eroaviksi mediatyypeiksi lasketaan tässä työssä normaalista poiketen:
- IP-puhelut (audio + video)
- PSTN-puhelut (audio)
- Datana liikkuvat sanomat (esim. chat, sähköposti)
- Muut myöhemmin esiteltävät tyypit, joita ei ole järkevää muuntaa miksikään em. mediatyypeistä

- em. järjestelmässä käytettävien standardien ja suositusten tarkasteluun ja vertailuun lähinnä niiltä osin, kuin ne ovat järjestelmän kannalta merkityksellisiä.

Ensimmäinen ongelmakentän osa on mediariippumattoman "puhelun"ohjausjärjestelmän teorian rakentamista ja pohtimista, erityisinä osinaan siltä odotettu logiikka (toiminnallisuus) ja järjestelmän ohjaamien komponenttien identifioiminen, sillä järjestelmän olisi oltava mahdollisimman joustava ja yleinen mutta samalla riittävän tiivis ollakseen käyttökelpoinen. Järjestelmän teoriaan kuuluu niinikään myös sen arvioitu arkkitehtuuri komponentteineen ja näille kuuluvine ominaisuuksineen.

Toisena osana oleva järjestelmän käytännön toteutus ei ole mahdollista siinä laajuudessa kuin se on koko järjestelmää koskevassa teoriassa esitelty. Käytännön toteutus paneutuu nimenomaan IP- ja PSTN-maailmoissa liikkuvan audio- ja videovirtojen keskitettyyn ohjaamiseen. Tässä perusongelmana on paitsi edellä mainittu PSTN- ja IP-verkkojen erilainen filosofia, myös käytettyjen standardien yhteensovittaminen.

Kolmas osa on verraten itsenäinen kahteen ensimmäiseen osaan nähden: se on perustavaa laatua oleva ongelmakenttä puhelunohjaukseen käytetyistä suosituksista ja standardeista, josta on ratkaistava riittävä osa, jotta voitaisiin toteuttaa USQ:n esimerkkitapaus. Tarkemmin ottaen kyseeseen tulevat lähinnä ECMA:n CSTA-mallin ja ITU-T:n H.323-standardin yhteensovittaminen niitä käyttävien rajapintojen (Microsoftin NetMeeting API ja TAPI 3.0) ja sovellusten kautta.

Koska ongelmakentän eri osien ratkaiseminen ei ole mahdollista täysin toisistaan riippumattomalla tavalla, esitetään osat siinä järjestyksessä, kuin vaaditut esitiedot kussakin tapauksessa vaativat: käytetyt standardit ovat yleisiä eivätkä ota suuremmin kantaa niitä käyttäviin järjestelmiin. Koska standardeihin vaaditut esitiedot eivät kuulu enää tämän työn alueeseen, esitetään standardit muita edellä. Standardit toimivat pohjana USQ:n teorialle, joka esitetään seuraavaksi. Näistä molemmista riippuva käytännön toteutuksen ratkaisu, HM&V Telecommunications Oy:n puhelunohjausjärjestelmän, CustomerConnect 2.0:n IP-yhteys esitellään viimeiseksi. Tähän läheisesti liittyen on lopussa myös eräitä liitteitä itse ohjelmatyöstä.

1.3 Rajaukset

Pohjana ja laajennuksen kohteena olevana PSTN-ohjausjärjestelmänä on käytettävissä HM&V Telecommunications Oy:ssä kehitetty ohjelmistopaketti, CustomerConnect. Vastaavasti IP-ohjelmistokomponenttina käytetään H.323-standardia seuraillen ohjelmallisesti ohjattua Microsoftin IP-puheluihin tarkoitettua NetMeeting-sovellusta. CustomerConnect toimii CSTA-mallin ja TAPI 2.1:n pohjalta [HM&V98], kun taas NetMeeting H.323:n ja oman rajapintansa, NM-API:n kautta [MICR97b].

Standardien selvityksessä rajoitutaan tutkimaan CSTA- ja H.323-standardeja, ensimmäistä CustomerConnectin käyttämässä laajuudessa ja jälkimmäistä NetMeetingin ja yleistetyn kontrollointijärjestelmän puitteissa. H.323:n puolesta tähän eivät kuulu muut H.225-viestit kuin liitteessä B esitetyt puhelusignalointikanavaan kuuluvat viestit, ja CSTA:n kohdalta mukaan lukeutuvat ainoastaan S-domainin tarjoamat palvelut, ja jotkin SR-domainin VRU-yksikön (eräänlaisena automaattisena puhelinvastaajana toimiva Voice Response Unit) palveluista. Vertailun vuoksi käydään läpi telealan määrittelemä (mm. Bell, Nortel, Pacific Telephony Design) Simple Computer Telephony Protocol (SCTP), ja selvitetään sen erot H.323:een ja miksei sitä ole valittu toteutusprotokollaksi.

Standardiselvitykseen lukeutuvassa rajapintojen selvityksessä käsitellään NetMeetingin ohjelmointirajapinta puhelunohjauksen kannalta ja TAPI 3.0 lyhyesti tulevaisuuden ohjelmointirajapintana CT-tekniikoille. Näistä NetMeeting-API:n tiedostonsiirto- ja hakemistopalvelujen käyttörajapinnat eivät kuulu tämän työn alueeseen ja toisaalta TAPI 3.0 ei vielä toistaiseksi julkaisemattomana rajapintana kuulu tähän työhön muuta kuin jatkokehitystä ohjaavana ohjenuorana.

USQ:n teoriasta käsitellään yleinen arkkitehtuuri ja sovelluslogiikka sekä joitakin komponenteilta vaadittavia ominaisuuksia. Näistä käydään tarkemmin lävitse vain IP-PSTN-logiikan vaatimat komponentit ja arkkitehtuurit. Erityisesti komponenttien rakenne esitellään ainoastaan H.323:n mukaisten osien kohdalla. USQ:n logiikan ja toiminnallisuuden rajaukset käydään tarkemmin lävitse USQ:n määrittelystä kertovassa luvussa.

USQ:n toteutuksen teoriasta tähän työhön kuuluvat yhteyksien ja puuttuvien komponenttien rakentaminen terminaalista gatekeeperiin tulevaan liityntään saakka. Itse gatekeeperin logiikkaa ei käsitellä muuten kuin USQ:n teoriaan liittyvin osin.

Tämän työn ohjelmointiosuus muodostuu osin eräiden valmiiden ohjelmistokomponenttien ja osin itsetehtyjen komponenttien ohjelmoimisesta ja yhteenliittämisestä. Tarkoituksena ei kuitenkaan ole kuvata valmiiden komponenttien rakennetta enempää kuin on välttämätöntä itse kokonaisuuden kannalta: esimerkiksi NetMeetingistä esitellään vain tämän ohjelmointirajapinta ja sen mukautuminen H.323:n spesifikaatioihin. Yhdyskäytävästä käydään lävitse niiden toiminnan ja ohjauksen vaatimat edellytykset mutta ei puututa niiden audiokoodauspuoleen.

On huomioitava, että tämä työ keskittyy nimenomaan multimediavirran *ohjaukseen*, Se siis ei käsittele mediavirran koodaukseen liittyviä algoritmeja ja tekniikoita, vaan ohjausdatavirran problematiikkaa. Tällöin esimerkiksi koodekit ja kykyjenvaihto (= mediavirran päätepisteiden välillä lähetetyt tiedot tietokoneen ominaisuuksista käsitellä mediavirtaa) jätetään mediavirran päätepisteiden huoleksi. Samoin alla käytettyjen siirtoteiden fyysinen toteutus ja protokollat sekä muiden kuin ohjemistokomponenttien tekninen arkkitehtuuri jäävät diplomityön aihepiirin ulkopuolelle. Itse ohjausjärjestelmän

ulkopuolelta ainoastaan siihen liittyvien komponenttien rajapinnat ja toiminnallisuus kuuluvat tämän työn alueeseen, ja nekin vain siltä osin, kuin muun selvityksen kannalta katsotaan välttämättömäksi.

2. USQ:n alusta ja lähtökohdat

2.1 Yleistä

USQ:n implementointia varten tarvitaan sekä tietojenkäsittelylaitteistot ohjelmistoineen että liittymät PSTN-ympäristöön. Tietojenkäsittely-ympäristön pohjana ovat multimedia-PC:t joissa ajetaan Microsoftin käyttöjärjestelmiä, joko Windows NT Server 4.0:a (Service Pack 3), tai Windows NT Workstation 4.0:a. Tietoverkkoalustana on TCP/IP-protokollaa käyttävä intranet/LAN (joskaan toteutus ei ole riippuvainen siirtotien nopeudesta). PSTN-puhelunohjausta varten on käytettävissä Siemens Hicom PSTN-vaihde (PBX) ja CTI-ohjelmistona HM&V Telecommunication Oy:ssä tehty middleware-ohjelmisto CustomerConnect, jonka nykyistä versiota 1.5 on tarkoitus käyttää USQ:n toteutuksen "emo-ohjelmistona". (ts. PSTN-puhelunohjausjärjestelmä laajennetaan USQ-järjestelmäksi lisäämällä siihen vähintään yksi uusi mediatyyppi käsiteltäväksi.)

2.2 Internet

Internet on pakettivälitteisten tietokoneverkkojen ei-triviaali joukko. Alimpana käytettynä protokollana toimii IP. Verkkojen välisessä tietoliikenteessä tämän päällä ajavat muut protokollat; ainoastaan aliverkon sisäisessä liikennöinnissä saatetaan käyttää esimerkiksi NetBEUI:tä tai IPX/SPX:ää [STAL94]. Nykyisin Internetin tunnetuin käyttömuoto on WWW, World Wide Web, jossa yhteydet koneiden välillä muodostetaan korkeamman tason protokollien avulla ja tietokoneiden julkistamat dokumentit ja aineisto on koottu standardoidulla tavalla koodattuihin tiedostoihin. Tärkeimpiä tässä työssä tarvittavia Internetiin liittyviä käsitteitä ovat: URL, eli Uniform Resource Locator, jonka avulla dokumentit ja tietokoneet voidaan identifioida; SGML-pohjainen HTML, eli HyperText Markup Language, paljon elänyt ja muuttunut standardi koodaustapa kunkin tietokoneen julkistamille dokumenteille; ja socket, kahden tietokoneen välisen loogisen yhteyden toinen pää. Tässä dokumentissa tarvitut muut tärkeät käsitteet on listattu alla.

2.2.1 WWW:ssä käytettyjä ohjelmistokomponentteja

WWW oli alunperin tarkoitettu eri puolilla maailmaa sijaitsevien teknisten dokumenttien vaivattomaan katselemiseen ja selailuun. Tältä pohjalta ovat myös kehittyneet tärkeimmät WWW:ssä liikkumiseen käytetyt ohjelmat / ohjelmistot: palvelimet ja selaimet.

Web-serveri (palvelin) on tietokoneessa se prosessi, joka kuuntelee yhteydenottopyyntöjä IP-yhteyden kautta tietyssä hyvin tunnetussa portissa, ja vastaa niihin lähettämällä tiedostoja tai muuta tietoa hakijan prosessille. Palvelin kommunikoi muiden prosessien kanssa tietyn protokollajoukon välityksellä, missä käytetty protokolla

riippuu tarjotusta palvelusta (WWW, FTP (*), Gopher, etc) ja palvelun pyytäjästä (selain- / palvelinkoneessa suorittava prosessi). WWW-palvelussa tärkeimpänä protokollana on HTTP (HyperText Transfer Protocol), joka on suunniteltu HTML-muotoisten ja sen sukuisten dokumenttien lähettämiseen Internetin ylitse.

Selain on tietokoneen se prosessi, joka lähettää yhteydenottopyyntöjä oman IP-yhteytensä kautta Internetin muihin tietokoneisiin HTTP-protokollaa käyttäen. Selaimen tehtävänä on toimia käyttäjärajapintana WWW:ssä liikkumiseen tulkitsemalla HTML-formaatissa olevat dokumentit näkyvään muotoon. HTML:n lisäksi on kehitetty useita keinoja lisätä dokumenttien interaktiivisuutta, mitä varten selaimissa on lisäksi usein valmius erityypisten ohjelmien suorittamiseen. Ohjelmat on yleensä tehty jonkin selaimen tukeman skriptauskielen tai välikoodiksi käännetyn tulkattavan kielen avulla. Selain voi myös käyttää hyväkseen tietokoneessa olevia muita ohjelmia tietynlaisen sisällön näyttämiseen tai ajamiseen. Välikoodiksi tulkattavalla kielellä tarkoitetaan tässä lähinnä Javaa, jonka yksinkertaistettu muoto JavaScript on esimerkki em. skriptauskielistä.

2.2.2 WWW:ssä käytettyjä ohjelmointikieliä

Dokumentit kirjoitettiin aluksi HTML-spesifikaatioiden mukaan, jonka pohjalta yleisimmät selaimet on rakennettu. WWW:n levittyä alkuperäisestä tarkoituksestaan laajemmin suuren yleisön käyttöön sen sisältämiltä dokumenteilta vaadittiin yhä monipuolisempaa toiminnallisuutta, mikä puolestaan johti CGI:n käytön lisääntymiseen. CGI (Common Gateway Interface) on yksinkertainen mekanismi, jossa selain pyytää palvelinohjelmistoa käynnistämään toisen prosessin palvelintietokoneessa suorittamaan jotakin selainkoneen määräämää tehtävää. Valitettavasti kytkeytyneiden selainten määrän kasvaessa suureksi palvelinkoneen resurssit alkoivat olla kriittisen vähissä. Oli tarvetta kehittää uusia keinoja palvelupyyntöjen suorittamiseksi: aluksi säiepohjainen suoritusmetodi palvelimeen (esim. ISAPI, NSAPI), ja pikkuhiljaa koko taakan siirtäminen muualle [mm. WEBB96].

Aluksi siis nopeuden ja myöhemmin myös reaaliaikaisuuden takaamiseksi (esim. valvonta- ja ohjausjärjestelmät) oli kehiteltävä keinoja saattaa logiikka osin selainkoneen suoritettavaksi. Nyttemmin näitä tapoja on useita: Java, JavaScript, JScript, VBScript, Dynamic HTML, ActiveX, yms. Tässä dokumentissa tarvitaan näistä kolmea käsitettä: ActiveX, Java ja VBScript.

Java on C++:sta muokattu lähes täysin objektiorientoitunut lausekieli, josta kääntäjät tuottavat hyvin kompaktia metakoodia, jota kussakin ympäristössä oleva nk. Java Virtual Machine tulkkaa. Javan tulisi olla alustariippumaton, mutta se ei oikeastaan tarkoita muuta kuin konekielitason tulkkauksen siirtämistä alemmalle tasolle, ts. virtuaalikoneelle.

(*) Nykyään FTP-käsite on siirtynyt protokollasta sitä tottelevien tiedostopalvelimien tarjoaman palvelun nimeksi.

Internet-teknologiassa Java-kääntäjällä tehdyllä koodilla on pienuutensa ja ilmaisuvoimansa vuoksi suuri merkitys.

Javalla voi tehdä sekä itsenäisiä ohjelmia, että appletteja, "sovelmia". Appletit ladataan HTML-koodin seassa olevien linkkien avulla erillisinä tiedostoina palvelinkoneelta, ja suoritetaan selaimen VM:ssä osana selainprosessia. Appletteihin tulee ohjelmoijan kirjoittaa tiettyjä entry-pointteja selainprosessille, joiden kautta selaimen koodista siirrytään appletin koodiin.

Javan turvallisuus on nk. hiekkalaatikkomallin ansiosta hyvä, ts. palvelimelta ladattu ja selaimessa ajava java-ohjelma ei pääse selaimen muistiavaruuden ulkopuolelle. Tämä turvallisuus on kuitenkin käytännössä paljolti selaimen määrättävissä, koska eräitä asioita ei yksinkertaisesti voi tehdä, mikäli turvallisuus otetaan liian vainoharhaisesti. Periaatteessa applet ei voi esimerkiksi kirjoittaa tai lukea selainkoneen levyä ilman erikoisia järjestelyjä, mutta myös tiettyjen ohjelmien käyttö appletista käsin on eräissä selaimissa kokonaan kiellettyä.

Java-appletteja voi kehittää useissa eri kehitysympäristöissä, nykyisin lähinnä kahden java-version perusteella. Sunin alunperin lisensoima ja standardoima java-kieli, jonka kehitysympäristö kulkee Java Development Kit-nimellä (JDK), ei ole enää ainoa javasta tehty toteutus [SUN98]. Microsoftin oma java-versio – toteutettu SDK for Java:ssa – eroaa Sunin versiosta eräiltä osin. Tässä dokumentissa käsitelty toteutus käyttää MSVJ++ 1.1:ä, joka pohjautuu JDK 1.0.2:een.

Javan käyttötarkoituksena on tässä työssä useiden eri sovellusten liimaaminen yhteen käyttämällä kullekin ominaista rajapintaa.

ActiveX on Microsoftin käyttämä hajautettujen sovellusten toteutukseen tarkoitettu käsite. Tässä dokumentissa ActiveX-komponenteilla viitataan tietyt rajapinnat ja käyttäytymismallin omaaviin paikkariippumattomiin ohjelmistokomponentteihin. Kyseiset ohjelmistokomponentit voi kirjoittaa usealla eri ohjelmointikielellä, kuten VisualBasicillä, C++:lla ja javalla, mutta niiden toimivuus muissa kuin Microsoftin järjestelmissä on epävarmaa. Koska tässä työssä toteutus on tehty nimenomaan Microsoftin ympäristöissä, ActiveX tarjoaa erinomaisen tavan käyttää tietokoneiden resursseja WWW:n ylitse.

ActiveX-komponentteja voi "upottaa" (embed) HTML-dokumentteihin java-appletin linkkien tapaan. Kyseisiä komponentteja ei tässä tapauksessa kuitenkaan ladata palvelimelta, vaan ne on oltava valmiiksi asennettuina selaimen järjestelmissä. Jollei komponenttia löydy asennettuna, selain voi tarjoutua hakemaan sen kohdatessaan kyseistä komponenttia vaativaa sisältöä.

ActiveX-komponenttien turvallisuusajattelu on hiukan erilainen kuin applettien: sen sijaan, että komponenttia estettäisiin tekemästä vahingollisia operaatioita, varmistetaan kaksi asiaa: varmistetaan, että vastuu on käyttäjällä antamalla käyttäjälle mahdollisuus päättää,

suoritetaanko mahdollisesti turvallisuudelle vaarallista koodia; ja lisäksi varmistetaan, että komponentin kirjoittaja jää kiinni, mikäli hänen ohjelmansa suorittaa selainkoneelle vahingollisia toimenpiteitä. Avainsanana ovat ns. sertifikaatit, jotka epäsymmetrisiä salausmetodeja (RSA) käyttäen liittävät ohjelman binääritiedostoon sen tekijän digitaalisen nimikirjoituksen. Kun ActiveX-komponenttia otetaan käyttöön, selain kysyy, luottaako käyttäjä sen mukana olevaan sertifikaattiin, mikäli sellaista on. Tämän jälkeen on käyttäjän harkinnan varassa, haluaako hän ajaa sivun mukana latautuneen ohjelman.

Tässä työssä käytetyt jonotiedote- ja IP-puheluohjelmat ovat käytännössä ActiveX-komponentteja, ja toimivat USQ:n ”työjuhtina” varsinaisen ohjauslogiikan sijaitessa muualla.

VBScript on Visual Basicistä tehty skriptausversio. Tämä tarkoittaa, että rajoitettua VisualBasic – koodia (VBScript on vain osajoukko koko VisualBasic-kielestä) voi kirjoittaa sellaisenaan HTML-sivulle, ja jos käytetty selain ymmärtää kyseistä skriptaus-kieltä, se tulkkaa skriptin ja suorittaa siinä annetut komennot. VBScript on ohjelmointikieli, ja sen koodi ladataan sellaisenaan HTML-sivun mukana. Sen tarjoama turvallisuus on lähempänä Javan kuin ActiveX:n turvallisuutta. Turvallisuuden perustana on VisualBasiciä pienempi ilmaisuvoima ja toisaalta lähdekoodin luettavuus. (Kuka tahansa pystyy tarkistamaan ohjelman toiminnan katsomalla sivun HTML-koodia)

VBScriptiä voi käyttää muiden selainpään skriptauskielien korvikkeena, tai ohjaamaan ActiveX-komponentteja. Tässä työssä jälkimmäinen ominaisuus on perusteellisemmin hyödynnettynä kuin ensimmäinen: johtuen ActiveX-komponenttien yleisimmän toteutus-kielen C++:n ja javan eroista, ei tiettyjä ActiveX-komponenttien ohjaukseen liittyviä asioita voi tehdä suoraan javasta käsin, vaan väliin tarvitaan pieniä VBScriptillä kirjoitettuja osuuksia (tästä enemmän liitteessä A).

2.3 Puhelinverkko ja puhelunohjaus

Public Switched Telephone Network, eli PSTN-käsite abstrahoidaan tässä toteutuksesta siten, että tarkastellaan ennemminkin verkossa kulkevaa mediavirtaa (audio- ja mahdollisesti videovirtaa). Tällöin PSTN-maailmaa esittää ajan suhteen muuttuva verkko, jonka solmuina ovat päätepisteinä toimivat puhelinterminaalit ja ohjauspalveluja tarjoavat kytkentäkomponentit. Kaaret ovat muodostettuja yhteyksiä yhdestä solmusta toiseen, jota myöten voi kulkea moduloitua ääntä tai erikoistapauksissa videota. Kytkentäkomponentit kuuluvat usein yhteen kymmeniä tai jopa tuhansia solmuja sisältävään kokonaisuuteen, jota kutsutaan vaihteeksi tai keskukseksi sen toimintaperiaatteen ja koon mukaan. PSTN-maailmaan lasketaan kuuluvaksi myös sen ohjauslogiikka, jota tarjoavat verkon alijoukkoina toteutetut älyverkot (tyypillisesti joukko palvelintietokoneita ja kontrolli-pisteitä), sekä pienemmät vaihdekomponentit, kuten PBX:t, integroidut piirit ja näiden yhteydessä mahdollisesti toimivat CT-ohjelmistot. Joskaan tässä dokumentissa esitetyt

standardit ja osa USQ:n teoriaa eivät ota kantaa PSTN:n tai muun mediatyypin komponenttien laatuun tai kokoon, on kuitenkin implementointia ajatellen tarkoituksenmukaisinta keskittyä sen käyttämään ympäristöön, eli tavalliseen puhelinvaihteeseen ja sen ominaisuuksiin. Niinpä tässä luvussa ei paneuduta PSTN:n määrittelyä syvemmin varsinaisiin puhelikeskuksiin ja älyverkkoihin: riittävä tieto näistä on se, että vaihteille voidaan tarjota jonkinasteista esiohjausta jo älyverkosta käsin [HANN96].

Vaihteella tai PBX:llä tarkoitetaan tässä sellaista päätelaitetta, josta käsin pystytään joko sisäisin määrittelyin tai ulkoisesti komentaen ohjaamaan tietyn kokoista puhelinterminaalien joukkoa. Vaihde voi olla joko kytkentämatriisi- tai väyläpohjainen, kunhan sillä pystytään toteuttamaan joukko toimintoja ACD:stä yksittäisen käyttäjän tavoitettavuuteen [HUOM93].

Puhelinterminaali on PSTN:stä puhuttaessa mikä tahansa keskukseen suoraan tai vaihteeseen kytkeytymään kykenevä laite, jonka tarkoituksena on siirtää informaatiota. Tässä kuitenkin tekstimuodossa liikkuvaan dataan tarkoitetut laitteet, kuten faksit, jäävät vähemmälle huomiolle pääpainon ollessa äänen- ja kuvansiirtoon tarkoitetuissa laitteissa (pääasiassa mobiili- ja lankapuhelimet)

Periaatteessa IP-puhelimilla pystytään jo nyt tekemään samoja asioita kuin tavallisilla puhelimilla, jos puhutaan yksityiskäytöstä, ts. tapauksesta, missä sekä mahdollisia soittajia (A-tilaajia) on täsmälleen yksi, samoin kuin puhelun mahdollisia vastaanottajia (B-tilaajia). Asia, mihin IP-puhelut eivät toistaiseksi ongelmitta sovellu, on monimutkaisempi puhelunohjaus, eli reitit ykset ja ACD [HANN96].

ACD, eli Automatic Call Distribution on termi jolla tarkoitetaan väljästi ottaen järjestelmään sisääntulevan puhelun ohjauslogiikkaa ja sen toteutusta puhelinterminaalien osoitteita korkeammalla abstraktiotasolla. Tähän kuuluu mm.

- Puhelinlaitteiden abstrahointi henkilöiksi tai käsitteiksi kuten palveluryhmiksi pitämällä yllä tiettyjä ajantasaisia assosiaatioita kunkin käsitteen ja puhelinlaitteen välillä
- Ruuhkatilanteiden käsittelyn monipuolistaminen, mm. jonotuksen, ylivuodon käsittelyn ja ruuhkan tasapainottamisen avulla
- Raportointi ja puheluiden valvonta ylipäätään
- Tehostettu hallinta

Tämän työn puitteisiin kuuluvat lähinnä kaksi ensimmäistä kohtaa, joista ensimmäisen kohdan palveluryhmän tavoitettavuutta käytetään toteutuksessa esimerkkinä sen suhteellisen monipuolisen luonteen vuoksi.

Seuraavassa esitellään yleisiä PSTN-puolella nähtävissä olevia ACD-ominaisuuksia, joihin tässä työssä tehtävä USQ:n toteutus pyrkii lisäämään uuden median (taulukko 2.1):

Taulukko 2.1: yleisiä ACD-ominaisuuksia

Palveluryhmät	Näihin voi yksittäinen käyttäjä, "agentti", liittyä ja poistua dynaamisesti. Riippuen ryhmäkuuluvuuksista agentille tarjotaan jollakin todennäköisyydellä sisääntulevia puheluja. Ryhmä on vaihteelle yksi looginen numero, jonka päässä ei ole puhelinlaitetta, vaan on pelkkä muistipaikka vaihteessa, johon puhelu on kytkettynä niin kauan kuin löydetään varsinainen numero.
Puheluiden jono	Yleisessä tapauksessa puheluita on tulossa enemmän kuin on niihin vastaamaan kykeneviä agentteja. Tällöin puhelu jonotetaan. Kun puhelu on jonossa, se voidaan asetusten perusteella kytkeä joko tavalliseen jonopaikkaan (kuten looginen numero, mutta useampi varastossa) tai VRU-yksikön linjaan, missä tapauksessa soittaja kuulee valintaäänen sijasta kyseiselle linjalle määriteltyjä tiedotteita tai musiikkia. Puhelu siirretään pois jonosta jonologiikan mukaisesti
Jonologiikka	Puhelut ovat yleensä jonossa FIFO-periaatteella, mutta tietyissä tapauksissa, joista enemmän USQ:n määrittelyssä, voidaan uusimpia puheluita laittaa keskelle jonoa tai poistaa joitakin puheluja käsittelystä ennen edellä olevia. Poikkeavaa logiikkaa käytetään mm. A-numeron perusteella tehdyn asiakastunnistuksen yhteydessä ja nk. taitopohjaisessa reitityksessä [HANN96]. On huomattava, että puhelujonon FIFO-periaatteen väljyys riippuu paljolti myös ns. tarjontalogiikasta.
Tarjontalogiikka	Tarjontalogiikalla tarkoitetaan sitä periaatetta, jonka mukaan jonosta tarjotaan puheluita agenteille vastattavaksi. Ryhmä voi olla joko "offer-to-all"- tai "offer-to-one"-tyyppinen."Offer-to-one" ryhmässä/jonossa puhelua tarjotaan vain yhdelle agentille, joka yleensä on esim. pisimpään joutilaana ollut agentti. Jono ei etene, ennen kuin valittu agentti on vastannut, tai logiikka vaihtanut kokeiltavaa agenttia. Voidaan lisäksi määritellä, kuinka toimitaan, jos agentti ei vastaa: siirrytäänkö seuraavaan, ylivuodetaanko, vai luovutetaanko kokonaan.

	"Offer-to-all"-ryhmässä kaikki agentit näkevät pääkäyttäjän määrittelemän osan jonosta. Tästä jononäkymästä kukin voi poimia mieleisensä puhelun. Yleensä ryhmäkohtaisia jonoja kohdellaan yhtenevästi, ts. Useampaa periaatetta ei normaalisti sovelleta samaan jonoon tai ryhmään. Tarjontalogiikka on toinen asia, jota USQ muuttaa melko radikaalisti.

ACD-ominaisuudet ovat vahvasti vaihdekohtaisia, ja voidaan toteuttaa paitsi vaihteen sisälle integroituina piireinä, myös vaihteen ulkopuolella olevalla ohjelmistopohjaisella ACD:llä. Seuraavassa esiteltävän USQ:n "emo-ohjelmiston", CustomerConnectin päätehtävä on nimenomaan nk. software-ACD:n toteuttaminen PSTN:lle [HM&V97b]. (Itse asiassa USQ voidaan nähdä myös mediariippumattomana toteutuksena software-ACD:lle).

2.4 CustomerConnect - puhelunohjausjärjestelmä

CustomerConnect® on HM&V Telecommunications Oy:n kehittämä middleware-sovelluspaketti puheluiden keskitettyä ohjausta, hallintaa ja raportointia varten [HM&V97b]. Tämän työn lähtökohtana käytetty versio oli 1.5 (release), johon USQ-toiminnallisuus oli tarkoitus integroida. CustomerConnect 1.5:n rakenne on client-server pohjainen, ja sen toteutusympäristö on Windows-koneita sisältävä lähiverkko, jossa yhteydenpitoprotokollana on joko NetBEUI tai TCP/IP. Serveriä voi ajaa joko Windows NT 3.51 Server – tai 4.0 Server/ Workstation-käyttöjärjestelmien päällä. Client ajaa em. lisäksi myös Windows 3.1:n ja Windows 95:n päällä. [HM&V98] CustomerConnectista esitellään tässä tarvittava perustietous myöhemmin esitettävien tekstien syvällisemmäksi ymmärtämiseksi.

2.4.1 Puhelunohjauspalvelut ja muu toiminnallisuus

CustomerConnect toteuttaa joukon ECMA:n tietokone-ohjattua puhelunkäsittelyä käsittelevässä CSTA-mallissa määriteltyjä puhelunohjauspalveluja [HM&V97b]. Mediariippumattomuuden aikaansaaminen edellyttää, että vastaavat palvelut pystytään ennemmin tai myöhemmin toteuttamaan myös muilla medioilla kuin PSTN:n kanssa. Toistaiseksi IP-puheluteknologioiden kypsymättömyyden vuoksi ei ole ollut mahdollista toteuttaa kuin osa näistä palveluista.

CustomerConnectin hallitsemat ohjauspalvelut ovat:

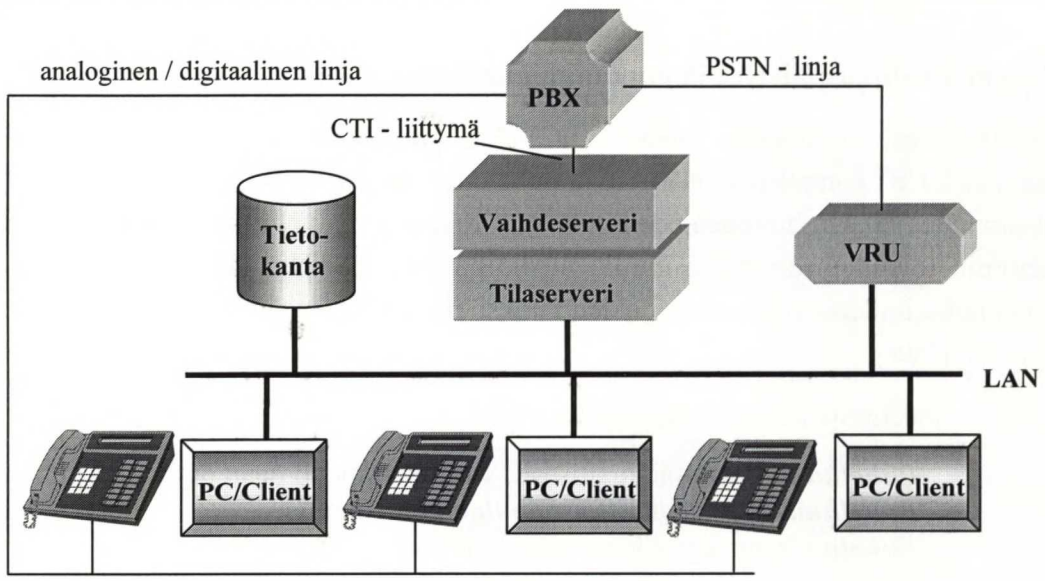
- ***Puhelun aloitus*** ohjelmallisesti, koskien nimenomaan outbound-puheluita
- ***Puheluun*** (hälyttävään) ***vastaaminen*** ohjelmallisesti
- ***Puhelun lopetus*** ja sen havaitseminen ohjelmallisesti

- ***Kutsunsiirto***, eli inbound- tai sisäpuhelun siirto hälyttämään toiseen laitteeseen, silloin kun se vielä hälyttää ensimmäisessä laitteessa
- ***Soitonsiirto***, eli käynnissä olevan puhelun siirto toiseen laitteeseen
- ***Puhelun*** siirto ***pitoon***, joka pitää yhteyden yllä, mutta katkaisee mediavirran
- ***Puhelun*** siirto ***pois pidosta***
- ***Konsultointipuhelu***, jossa käynnistetään toinen puhelu toiseen puhelinlaitteeseen, pitäen yhteys edelleen yllä edelliseen laitteeseen. Tässä uusi puhelu on aktiivisena ja vanha pidossa
- ***"Vaihtelu" (Alternate)***, jossa konsultointipuhelun pidossa ja aktiivisena olevat puhelut vaihtavat tilaa
- ***Palvelupuhelu***, eli agenttien osajoukon loogiseen numeroon tulevan puhelun reititys ja käsittely
- ***Henkilökohtaisten puheluiden reititys***
- ***Soittolistat***, eli lista henkilöistä, joihin on otettava yhteys, ja joille soitetaan pelkästään klikkaamalla kyseisen tahon tunnistetta.

Edellä lueteltujen puhelunohjauspalveluiden lisäksi CustomerConnectiin on toteutettu koko joukko liittymäpintoja mahdollisiin yritysjärjestelmiin, kuten asiakastietokantoihin. Raportointiominaisuudet, samoin kuin ryhmän (agenttien osajoukko) hallintaan ja pääkäyttäjän toiminnallisuuteen liittyvät ominaisuudet ovat myös melko ekstensiiviset [HM&V97b], mutteivät oikeastaan tämän tekstin asiaa.

2.4.2 Arkkitehtuuri

CustomerConnectin pääkomponentit ovat CustmerConnect-client ja CustomerConnect-server. CustomerConnect-server jakautuu kahteen loogiseen osaan, jotka kylläkin sijaitsevat samassa ajonaikaisessa ympäristössä, nimittäin vaihdeserveriin ja tilaserveriin [HM&V98]. Kuva 2.1 selventää tilannetta.



Kuva 2.1: CustomerConnect 1.5 arkkitehtuuri

Vaihdeserveri toimii abstrahointikerroksena fyysiseen vaihteeseen päin ja tilaserveri sisältää suurimman osan logiikasta ja toiminnallisuudesta. Client on tavallisen käyttäjän ja pääkäyttäjän työskentely-ympäristö, eli käyttöliittymä, josta on yhteys tietokantaan ja serveriin LAN:n lävitse. Puhelinlaitteet liittyvät käyttäjän (josta CSTA-mallin mukaisesti käytetään nimitystä *agentti*) työympäristöön, mutta ovat kytkettyinä vain vaihteen alanumeroihin, eivät siis suoraan tietokoneisiin. Tietokanta toimii asiakas- ja agentti-informaation sekä raportointidatan säilytyspaikkana. VRU, eli Voice Response Unit on erillisellä piirikortilla oleva laitteisto, jonka avulla pystytään soittamaan äänitiedotteita vaihteen linjoille. VRU:ta ohjaava ohjelmisto kytkeytyy LAN:n kautta CustomerConnect-serveriin mukailtuna client-ohjelmistona [HM&V98].

2.4.3 CustomerConnectin käyttämät rajapinnat

Kommunikointirajapinnat ovat kolmansien osapuolien standardeja ulkopuolisiin komponentteihin nähden, mutta sisäiset komponentit kommunikoivat suhteellisen alhaisen tason yhteyksillä, jotka ovat standardoituja ainoastaan verkon protokollatasolla. Rajapinnat on luetteloitu taulukossa 2.2.

Taulukko 2.2: CustomerConnect 1.5:n rajapinnat

Rajapinnan nimi	Rajapinnan sijainti	Käyttötarkoitus
ACL	PBX – vaihdeserveri	Vaihdeserveri ohjaa erityistapauksena Siemensin Hicom-vaihdetta sen tarjoaman valmistajakohtaisen rajapinnan lävitse.
TAPI 2	PBX – vaihdeserveri	Vaihdeserveri ohjaa tämän rajapinnan lävitse TAPI 2 – yhteensopivia vaihteita. Näitä ovat esim. Meridianin ja Ericssonin eräät vaihteet.
ODBC	Tietokanta – muut komponentit	ODBC on yleinen RDBMS-rajapinta,, jonka lävitse tilaserveri ja client keskustelevat tietokannan (SQL-server 6.5) kanssa.
CSTA-rajapinta	Tilaserveri-vaihdeserveri	Tilaserveri ja vaihdeserveri keskustelevat keskenään kahdella abstraktiotasolla. Korkeampi niistä käsittää CSTA-mallin mukaisilla tunnisteilla kommunikoinnin.
A-rajapinta	Client – Server	Alhaisen abstraktiotason rajapintana on suoraan verkkoprotokollien päälle rakennettu asynkroninen viestirajapinta, jota (tila)serveri käyttää kommunikoidessaan clienttien kanssa. Sisältää tietyn viestijoukon.
X-rajapinta	Tilaserveri – vaihdeserveri	Samantyyppinen rajapinta kuin A-rajapintakin, mutta sisältää eri viestijoukon.

3. Standardit, suositukset ja rajapinnat

3.1 ITU-T:n H.323-standardi

H.323 on ITU-T:n (International Telecommunication Union Telecommunication Standardization Sector) 1996 lopussa päättämä standardi kapeakaistaisten visuaalisten puhelinpalveluiden teknisistä vaatimuksista yhden tai useamman lähiverkon alueella tapahtuvasta pakettikytkentäisestä liikennöinnistä, joissa palvelunlaatu (QoS, Quality of Service) ei välttämättä ole taattu [ITUT96] (toteutus kuten UDP:ssä – joka onkin mediansiirrossa käytetty protokolla IP-maailmassa – eli virhetilanteessa paketteja ei lähetetä uudelleen). Tämä luku selvittää suppeasti H.323:n tärkeimmät osa-alueet. Termistö selitetään sitä mukaa kun tuntemattomia termejä ilmenee.

H.323 nojaa vahvasti muihin ITU-T:n määrittelemiin standardeihin, sellaisiin kuten H.245 (mediavirran kontrolli), H.225 (puhelukontrolli), H.26x (videokoodekit) G.72x (audiokoodekit) ym. Nämä standardit ovat yleensä alhaisen tason protokollia, joita H.323-yhteensopivien komponenttien on noudatettava. Itse H.323 ei näitä kuitenkaan määrittele – se on nk. sateenvarjosuositus, joka ei määrittele protokollia, vaan arkkitehtuurin. H.323 ei siis ota kantaa alhaisen tason verkkoprotokolliin, paitsi mitä tulee kommunikointiin SCN:n (= Switched Circuit Network, PSTN-verkko, l. perinteinen piirikytkentäinen puhelumaailma) kanssa. Alhaisin taso, jota H.323 käyttää, ovat RTP- ja RTCP-protokollat pakettikytkentäisessä verkossa ilmenevien viiveiden ja muiden laatuongelmien minimoimiseksi [ITUT96].

H.323:n toteuttavan systeemin on oltava yhteensopiva kolmen asian kanssa: käytetyt koodekit, ohjausprotokollat ja arkkitehtuuri [ITUT96]. Eräissä tapauksissa kaikkia osia arkkitehtuurista tai komponenttien tarjoamista palveluista ei tarvitse täyttää, jotta aiottu järjestelmä olisi H.323-yhteensopiva. (Esimerkiksi NetMeeting, joka on virallisesti H.323-yhteensopiva terminaali ja MCU, ei tue Gatekeeperiä. Tämän puutteen tässä työssä haettu järjestelmä itse asiassa pyrkii korjaamaan.)

3.1.1 Kanavat ja informaatiovirrat

H.323-systeemin perustana ovat ns. informaatiovirrat (stream). Nämä virrat ovat mediavirran yleistyksiä siinä mielessä, että myös ohjausdata käsitetään virtana [ITUT96]. Yksi tai useampi ohjausdatan virta siis ohjaa varsinaisia mediavirtoja. H.323 jakaa informaatiovirrat viiteen kategoriaan, joista kolme on varsinaisia mediavirtoja ja kaksi ohjausdataa varten. Jokainen virta kulkee omassa loogisessa "kanavassaan" (paitsi puhelukontrollisignaalit, jotka on jaettu kahteen kanavaan). Tämä kanavajako mahdollistaa erityyppisten virtojen erottelun, mistä on hyötyä heterogeenisen laiteympäristöjen tapauksessa. Tietokonepäätteet esimerkiksi tukevat yleensä videon vastaanottamista, mutta mm. audion laita on vielä varsin epävarmaa. Erikseen nimetyt mediavirrat ovat audio-,

video- ja datavirta. Jälkimmäinen tarkoittaa kaikkea sellaista sisältöä välittävää virtaa, joka voidaan esittää still-muodossa datana, esimerkiksi kuvat, faksit, tiedostot ja chat. Kaksi kontrollivirtaa ovat tietoliikenteen kontrollisignaalivirta ja puhelukontrollisignaalien virta. Tietoliikenteen kontrollisignaalivirta on H.245:n määrittelemä kontrolliviestien joukko mm. loogisten kanavien avaamiseen ja sulkemiseen mediavirroille ja terminaalien media-ominaisuuksien selvittämistä varten. Puhelukontrollisignalointi on puhelun ohjausta H.225.0:n määrittelemien viestien sekä RAS-viestien (Registration Admissions and Status) avulla.

Em. informaatiovirroista kaikki niiden käyttämät kanavat ovat oletusarvoisesti suljettuja, paitsi H.245-kanava, joka sekin on avoinna ainoastaan loogisessa mielessä [ITUT96], koska sitä kautta annetaan mediavirtaa kuljettavien kanavien aukaisukomennot. Kaikki mediavirtakanavat voivat olla joko yksi- tai kaksisuuntaisia, riippuen osallistuvien terminaalien ominaisuuksista (so. jokin terminaali voi kyllä kuunnella ääntä ja katsella videota, muttei voi itse osallistua konferenssiin muuten kuin chatin avulla)

Hyväksytyt videon koodekit ovat H.261 QCIF ja CIF, H.263 SQCIF, QCIF, CIF, 4CIF ja 16CIF (määritelty vastaavissa standardeissa). Käytetty formaatti ja bittinopeus määritellään miniminä kaikkien osapuolien tukemista maksimiarvoista näille parametreille kanavien aukaisun yhteydessä H.245-standardin mukaisesti. Videovirta sisältää myös sen alhaisen tason kontrollointiin tarvittavat signaalit. Videon tukeminen ei ole pakollista yhdellekään H.323-komponentille, ellei sen H.323:n ulkopuolella oleva toiminnankuvaus sitä vaadi.

Hyväksytyt audion koodekit ovat G.711, G.722, G.723, G.728, G.729 ja MPEG1 audio (määritelty vastaavissa standardeissa). Audiovirta sisältää myös sen alhaisen tason kontrollointiin tarvittavat signaalit. Toisin kuin videon määrittelyssä, käytetty formaatti on standardoitu H.225.0:ssa vakioksi, joten kaikkien H.323-yhteensopivien komponenttien on käytettävä samaa audiovirran formaattia. Audion tukeminen on pakollista kaikille terminaaleille, muttei muille (paitsi jos komponentin H.323:n ulkopuolinen toiminnankuvaus sitä vaatii).

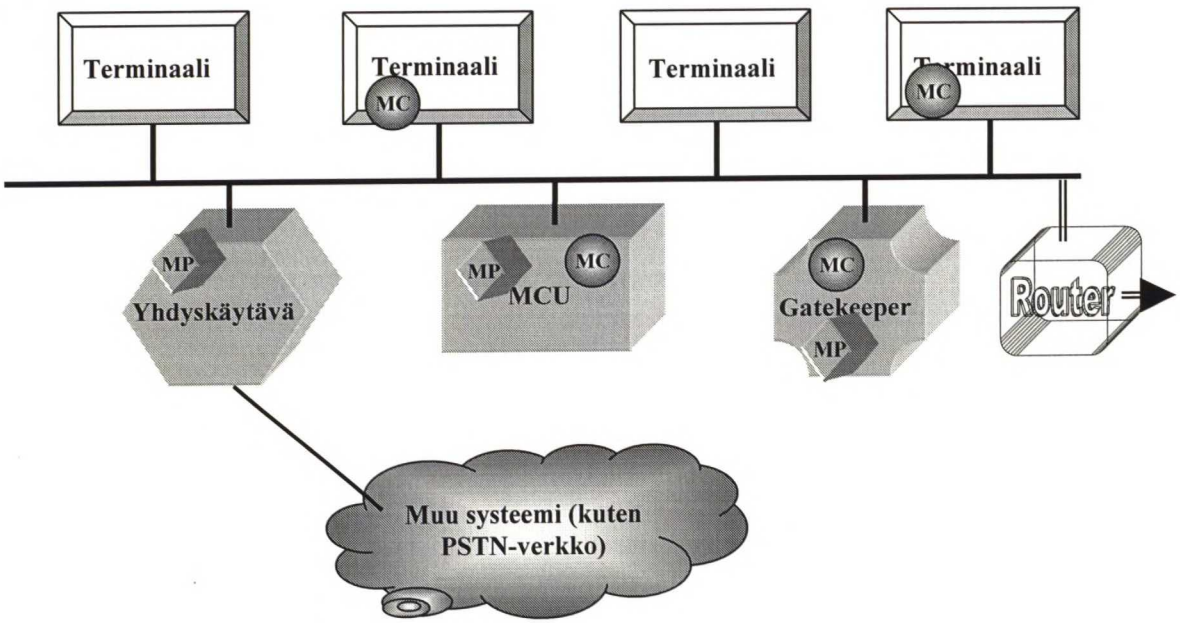
Datavirtaan käytetään pääasiassa T.120-standardia, ja poikkeuksina on vain eräitä etäohjattuja kameroita (H.323:een kuuluvina datavirtaa tuottavina elementteinä, jotka eivät kuitenkaan tue T.120:tä). Normaalisti mediavirran aukaisu ei onnistu ennenkuin H.323-puhelu/konferenssi on käynnissä. Datavirran tapauksessa alaspäin olevan yhteensopivuuden vuoksi T.120-kanavan aukaisu voidaan tehdä sellaisenaan, tosin tästä seuraa välittömästi H.323-puhelun luonti. T.120-kanava on muutenkin sikäli poikkeuksellinen, että sen kunnollinen toiminta vaatii kaksisuuntaisen liikennöinnin, ts. sille on luotava H.323:n puitteissa kaksi loogista yksisuuntaista kanavaa T.120-yhteyttä avattaessa. Datavirran tukeminen ei ole pakollista millekään H.323-komponentille, jälleen kuitenkin ainoastaan tämän standardin puitteissa.

Tietoliikenteen kontrollisignaalit, eli H.245-signalointi kulkee itsenäisenä informaatiovirtana omassa kanavassaan. H.245-kanavalle annetaan tunniste nolla, joka tarkoittaa, että loogisessa mielessä se on aina auki. Viestipohjaista H.245-kanavaa käytetään H.323-komponenttien yleiseen kontrollointiin, eli yhteyksien luomiseen, kanavien säätelyyn, eri päätepisteiden välisten ominaisuuksien selvittämiseen ja muihin sekalaisiin diagnosointi- ja hallinnointitehtäviin. Ominaista tälle informaatiovirralle on sen käyttötarkoitus enimmäkseen mediavirtojen hallinnointiin. Jokaista käynnissä olevaa puhelua kohden luodaan oma kontrollikanavansa, vaikka varsinainen mediavirta voi olla pelkkää datavirtaa (jokaisen terminaalin on *tuettava* audiota, mutta ei välttämättä *käytettävä* sitä), joten useita mediavirtoja tukevan komponentin on pystyttävä tukemaan useita H.245-kanavia. Kontrollisignalointi on standardoitu vastaavannimisessä suosituksessa (H.245), ja sitä standardin määräämänä tukevat komponentit ovat: terminaalit, gatekeeperit ja MCU:t. Huomattava on, että sisäinen MC voi standardin mukaan sisältyä myös yhdyskäytäviin, ja käytännössä usein sisältyykin terminaaleihin,joten käytännössä kaikki H.323-komponentit tukevat tätä informaatiovirtaa.

Varsinaiseen puhelunkontrolliin on olemassa kaksi signalointimenettelyä: RAS ja nk. puhelusignalointi (Call Signalling, H.225.0). Molemmat ovat viestipohjaisia ja kulkevat omissa itsenäisissä kanavissaan, joita ei hallinnoida H.245-kanavan kautta. Sekä RAS että puhelusignalointikanva aukaistaan ennen mediavirtakanavia kaikissa tapauksissa. RAS on tarkoitettu yksinomaan viestintään gatekeeperin ja terminaalin välillä, kun taas puhelunsignalointi hoitaa mm. puhelun aloittamis- ja lopetusfunktioita mediavirran päätepisteiden välillä. Sekä RAS:n että puhelusignalointikanavan viestit ja formaatti on määritelty H.225.0-standardissa, mutta niiden käyttö ja vastineet eri tilanteissa (lähinnä gatekeeperiä vastaan) määritellään H.323:ssa, muodostaen itse asiassa suurimman osan koko standardista.

3.1.2 H.323 - arkkitehtuuri

H.323 on yhteen tai useampaan pakettikytkentäisen verkon alueeseen (zoneen) määritelty suositus [ITUT96]. Vaikka se ei rajoitakaan komponenttejaan olemaan samassa alueessa, jokainen on silti kytkeytyneenä johonkin lähiverkkoon/intranetiin. Yhden alueen alueella olevat komponentit on esitetty kuvassa 3.1.



Kuva 3.1: H.323-systeemin arkkitehtuuri yhdessä alueessa

H.323-järjestelmässä on neljäntyyppisiä komponentteja: terminaalit, MCU:t (Multipoint Control Unit), yhdyskäytävät (gateway) ja gatekeeperit. Lisäksi tulevat alikomponentit: MC:t (Multipoint Controller) ja MP:t (Multipoint Processor). Kuvassa on edellämainittujen komponenttien rinnalla näytetty esimerkin vuoksi standardiin kuulumattomia elementtejä kuvaamassa H.323-järjestelmän yhteyksiä alueen ulkopuolelle, joita on yhdyskäytävän tarjoaman reitin lisäksi yleensä jonkin reitittimen lävitse kulkeva tie muihin lähiverkkoihin tai Internetiin. Alue on käytännössä pakettikytkentäisen verkon osajoukko, mutta suosituksessa se on yhtenevä yhden gatekeeperin alueen (=ne H.323-komponentit, joiden resursseja gatekeeper pystyy kontrolloimaan) kanssa [ITUT96].

Eri komponentit toimittavat kukin sille määriteltyjä tehtäviä H.323:n alueella: terminaali on mikä tahansa spesifikaatiot täyttävä puhelinlaite tai tietokonepääte, mediavirran tuottaja; yhdyskäytävä mahdollistaa nimensä mukaisesti yhteydet H.323:a tukemattomiin järjestelmiin, joista esimerkkinä mainittakoon PSTN-verkko; gatekeeper pitää huolta hallitsemansa Internetin alueen reititys- ja accessointimenettelyistä, kun taas MCU:n tehtävänä on toimia lähinnä konferenssien isäntänä ja huoltajana. MC on MCU:n kontrolliosa, ja MP taas mediavirran keskittävä elementti. Seuraavassa esitellään komponentit tarkemmin.

Terminaali on tärkeässä asemassa sikäli, että se toimii tavallisen käyttäjän komponenttina. Ohjelmistoteollisuudessa nämä ovat ensimmäisinä toteutettuja komponentteja, ja heikon alun jälkeen niitä alkaa olla useita suhteellisen luotettavia kaupallisessa tai freeware-jakelussa, esim. NetMeeting, sekä Intelin ja Netscapen IP-puhelimet [HANN96]. Terminaali

koostuu mediavirtoja tuottavasta laitteistosta, käyttöliittymästä, eri informaatiovirtojen käsittelyfunktioista ja koodekeista (esitelty edellä) sekä signalointirajapinnasta OSI-mallin verkkokerrokseen. Koko tästä toiminnallisuudesta H.323:n piiriin kuuluvat informaatiovirtojen koodekit, systeemin kontrollifunktiot sekä H.225.0-viestit informaatiovirtojen ja alemman tason verkkoprotokollien välissä.

Jokaisen terminaalin on pystyttävä audiovirran tukemiseen koodekin tasolla (vaikka laitteisto ei antaisikaan siihen mahdollisuutta) sen lisäksi se voi tukea video-koodekkejä ja T.120-datavirtaa, mutta ne eivät ole pakollisia. Kontrollikanavia (H.245, RAS ja puhelusignalointi) on jokaisen terminaalin tuettava. Esimerkkinä mediavirtojen koodekki-tuesta NetMeeting tukee oletusarvoisesti (versio 2.0) T.120:tä datan siirrossa, H.263:a videokonferensoinnissa ja G.723:a audion siirrossa. Terminaalin ei ole kuitenkaan pakko rajoittua H.323:n koodekkeihin: esim NetMeeting tukee oletuksena myös Microsoftin GSM-koodekkia [MICR97c].

Terminaalit voivat olla mediavirtayhteydessä toiseen terminaaliin, MCUhun tai yhdyskäytävään. Niitä komponentteja, jotka voivat olla mediavirtayhteydessä keskenään, kutsutaan päätepisteiksi. Päätepiste voi ottaa yhteyden toiseen päätepisteeseen joko suoraan tai gatekeeperin välityksellä. Suorassa yhteydenotossa päätepisteet ensin alustavat puhelu-signalointikanavan kautta puhelun sekä H.245-kanavan. H.245-kanavaa käyttäen osapuolet päättelevät toistensa mediaominaisuudet (capabilities exchange) ja ristiriitatilanteiden ratkaisumenettelyn (master-slave determination), ja sen jälkeen aukaisevat kunkin koodekin – ja käytännössä myös laitteiston – mahdollistamat mediavirrat, mikäli ne on määritelty aukaistaviksi käyttöliittymästä käsin.

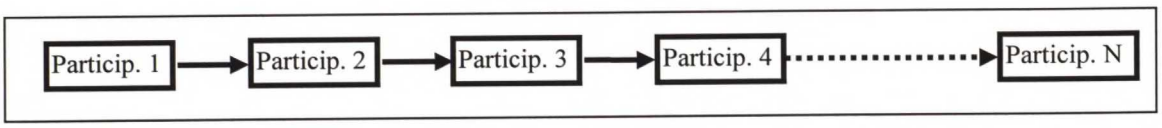
Jos alueessa on gatekeeper, molemmat päätepisteet rekisteröityvät ensin tälle, ja antavat sen päättää, kuinka toinen osapuoli haetaan ja kuinka puhelu yhdistetään. Rekisteröityminen, samoin kuin muukin kommunikointi gatekeeperin kanssa hoidetaan RAS-kanavan välityksellä. Valitettavasti gatekeeper on H.323-standardissa se osa, jota ei vielä yleensä tueta [HANN96]. Vaikka NetMeetingillä voi ottaa kontrolloidun yhteyden jopa H.323-yhdyskäytävään, gatekeeperiä se ei käytä, eikä siis myöskään RAS-kanavaa [MICR97b].

Terminaalissa voi olla MC, jotta ns. Ad Hoc-konferenssien (ts. alunperin kahdenvälisenä yhteytenä alkanut, mutta uusia jäseniä mukaan kutsumalla konferenssiksi laajennut puhelu) tukeminen olisi mahdollista (esim. NetMeeting sisältää MC:n). Vaikka MC yleensä on päätepiste, ei se terminaalin sisällä sitä ole. Ainoastaan terminaalin kautta voi MC:lle soittaa.

MCUn käsite edellyttää H.323:ssa esiteltyjen konferensointi- l. neuvottelumallien tuntemista. H.323 määrittelee kolme erityyppistä neuvottelupuhelun topologiaa:

- Hajautettu (Ad Hoc) malli, jossa mediavirta kulkee suoraan päätepisteiden välillä. Konferenssi syntyy, kun jokin päätepiste luo sen, ja siihen jo liittyneet päätepisteet kutsuvat mahdollisesti lisää jäseniä neuvotteluun.
- Keskitetty (Centralized) malli, jossa mediavirta kulkee keskitetyn komponentin kautta. Konferenssi syntyy, kun keskitetty komponentti ensin luo sen, ja kutsuu mukaan uusia päätepisteitä.
- Hybridimalli (Mixed), jossa on keskitetty ja hajautettu osa. Tässä keskitetyn osan keskittävä komponentti toimii linkkinä myös hajautetulle osalle olemalla yksi hajautetun konferenssiosan päätepisteistä.

MCU toimii usein konferenssi-isäntänä, l. sinä konferenssin osana, jonne kaikki ovat kytkeytyneet suoraan tai muiden MCU-Ad-Hoc-konferenssi-isäntien välityksellä. MCU:n tarve on ilmeinen, kun ajatellaan Ad-Hoc-konferenssin muodostumista (esim. NetMeetingissä): Koska varsinainen puhelukin on käytännössä aina konferenssi (joissa ei aluksi ole muita jäseniä kuin alkuperäinen konferenssin luoja), siihen kutsutut uudet jäsenet jäävät aina hierarkiassa kutsujan alapuolelle. Kutsutun näkökulmasta Ad-Hoc konferenssissa kutsuja on aina konferenssi-isäntä. Allaoleva kuva havainnollistaa äärimmäistä tilannetta:

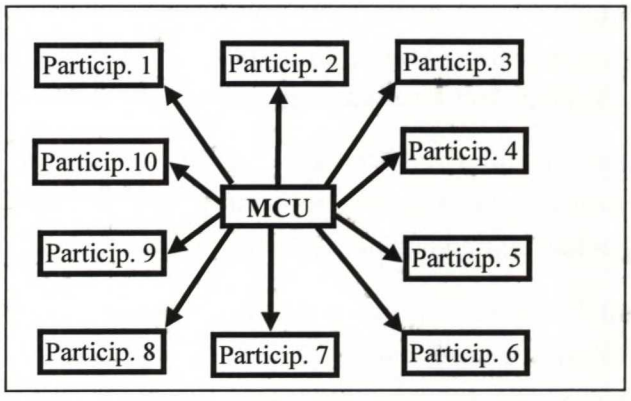


Kuva 3.2: Ad-Hoc konferenssi, joissa konferenssiin viimeiseksi kutsuttu toimii aina seuraavana kutsujana.

Ylläolevassa kuvassa konferenssin jäsen "Particip. k" on isännän suhteessa "Particip k+1":een nähden kaikilla k:n arvoilla. Koska koko konferenssi päättyy, kun isäntä poistuu, niin alkuperäinen kutsuja ei voi jättää konferenssia ennenkuin kaikki muut ovat poistuneet, koska muussa tapauksessa kaikki yhteydet katkeavat. (Huom. konferenssissa audio ei kulje kaikille, koska MP:tä ei ole. Sen sijaan esim. NetMeetingissä mediavirrat ovat point-to-point, joskin niiden päätepisteitä voidaan lennossa vaihtaa.)

Ideaalinen tapaus konferenssille olisi ns. keskitetty konferenssi, kuvattu kuvassa 3.3.

Keskitetyssä konferenssissa kuka tahansa isäntää lukuunottamatta voi poistua konferenssista muiden yhteyksien kärsimättä. Isännän rooli onkin tässä tärkeä: sen on oltava helposti saatavilla, eikä se voi oletusarvoisesti "olla pois päältä", ts. sen toiminnalliset keskeytykset on laskettava poikkeustilanteiksi.



Kuva 3.3: ideaalinen konferenssi, missä isäntä ei varsinaisesti osallistu konferenssiin

Keskitetyn konferenssin isännän rooli onkin MCU:n tärkein tehtävä. MCU:n on pakko sisältää MC konferenssin jäsenten hyväksymiseksi ja kutsumiseksi, mutta MP ei ole pakollinen. MP:n mukanaolo tekisi MCU:n roolin hyvin PSTN-maailman vaihdekomponentin kaltaiseksi. Itse asiassa, jos olisi olemassa kaupallisesti myytäviä middlewareksi tarkoitettuja (= suunniteltu vaihdetoimintaan kunnollisine ohjelmointi-rajapintoineen) MP:n sisältäviä MCU:ta, koko IP-puhelunohjauslogiikka muuttuisi radikaalisti. Tällöin ohjaukseen voitaisiin hyvin käyttää esimerkiksi CSTA-mallia. (esim RadVision, VideoServer ja OnLive! ovat kyllä H.323-yhteensopivia AV-puolen MCU:ta, mutteivät tarjoa middleware-kelpoisuutta, jota vaaditaan MCU:n istuttamiseksi mediariippumattomaan ohjausjärjestelmään.)

MCU:n on tuettava niitä mediavirtoja, joita se käyttää hyväkseen, mutta tämä on täysin parametroitavissa. Kontrollikanaviin pätee sama kuin terminaaleihin, onhan MCU yksi mediavirran päätepiste.

Yhdyskäytävän tehtävänä on tarjota H.323-järjestelmälle mahdollisuus kommunikoida muihin systeemeihin päin. Tärkeimpänä systeeminä on nimenomaan PSTN-verkko, jonne suunnatut yhdyskäytävät (eivät välttämättä H.323:a tukevia) ovat viime aikoina nousseet pinnalle eri laitetoimittajien implementointeina [HANN96].

Yhdyskäytävän käyttötarkoituksena on standardin mukaan esim. IP-puhelimesta (päätteeltä) soitto tavalliseen puhelimeen, eli yhdyskäytävä on se komponentti, joka tarjoaa lopullisen, todellisen mediariippumattomuuden. Ilman yhdyskäytävää eri tyyppisiä mediavirtoja voidaan ehkä kontrolloida yhtenäisesti, mutta niitä ei voida silti sekoittaa. Yhdyskäytävä tarjoaa myös epäsuorasti siirtotien valintamahdollisuuden. Koska sekä PSTN- että IP-liikenne kulkevat eri tyyppisesti osin jopa fyysisesti eri kaapelia pitkin, saattaa siirtotien valinta muodostua ratkaisevaksi tekijäksi: IP-soitot ovat pakettivälitteisiä, joten niiden käyttämät siirtotiet ovat eri tavalla hinnoiteltuja, mikä tekee IP-soitot tyypillisesti PSTN-puheluita halvemmiksi pitkillä matkoilla [HANN96]. Vastaavasti PSTN-

reitti on luotettavampi ja tarjoaa paremman laadum esim. audiossa. Tätä ominaisuutta voidaan käyttää hyväksi tarjottaessa korkeatasoisia multimediayhteyksiä.

Tärkeä yhdyskäytävän ominaisuus on, että siihen liittyvät järjestelmät näkevät sen tavallisena kyseiselle järjestelmälle ominaisena komponenttina. IP-PSTN-yhdyskäytävissä tämä tarkoittaa sitä, että H.323-puolen komponettien kannalta yhdyskäytävä käyttäytyy kuin terminaali tai MCU, ja PSTN-puolella sitä voidaan pitää joko tavallisena puhelimena tai vaihteena (PBX:nä). Toistaiseksi valmistetut yhdyskäytävät kattavat PSTN-puolella sekä jonkinasteisena vaihteena, että puhelinlaitteena toimivan komponentin funktionaalisuuden [PULV98], tosin niiden hyödyntäminen on käytännössä vaikeaa, koska APIt ovat yleensä epästandardeja, liian suppeita ja niiden käyttö vaatii lähes aina itse tuotteen ostamista.

H.323 määrittelee lisäksi yhdyskäytävän tarjoamat muunnospalvelut: yhdyskäytävän on pystyttävä muuntamaan paitsi mediavirta (ja tarvittaessa riisumaan siitä esimerkiksi videokanava pois, mikäli toinen puoli ei sitä tue) myös kontrollisignaalit. Erityisesti IP-PSTN-yhdyskäytävien tapauksessa piirikytkentäinen yhteys on voitava muuttaa pakettikytkentäiseksi ja H.225.0 (eli RAS ja puhelusignalointi) Q.931:n mukaiseksi signaalivirraksi tai vastaavaksi.

Gatekeeper on nimensä mukaisesti portinvartija, ja vaikka se ei olekaan pakollinen systeemissä, se tarjoaa hyvin pitkällevietyjä palveluja intranetin/alueen puheluyhteyksien keskitetyn hallinnan alalta. Gatekeeperin tehtäviin kuuluu mm. kaistanleveyden säätely (jonka strategia on suosituksen ulkopuolella), puhelujen määränpään reitittäminen, alueeseen oikeutettujen terminaalien seulominen, ja yleinen alueen (Zone) hallinta tarjoamalla ylläkuvattuja palveluja kaikille sille rekisteröityneille H.323-komponenteille [ITUT96] (muodostavat alueen). Alueessa ei kuitenkaan voi olla kuin täsmälleen yksi gatekeeper: useamman gatekeeperin tapauksessa myös alueita on useita, jos taas gatekeeperiä ei ole ollenkaan, järjestelmä ei muodosta aluetta.

Gatekeeper voi sisältää (kuten yhdyskäytäväkin) MC:n tai jopa MCU:n. Terminaalin lailla gatekeeperin MC on tarkoitettu vain Ad-Hoc-konferenssien tukemiseen, eikä siihen voi soittaa. Koska gatekeeperiin loogisena yksikkönä ei voi muutenkaan soittaa, ainoa tapa kutsua sen yhteydessä sijaitsevaa MC:tä on kutsua MCU:ta MC:n ympärillä [ITUT96], jos sellainen on.

Gatekeeperille on pakollisten yllämainittujen tehtävien lisäksi sälytetty joukko optionaalisia tehtäviä. Näitä on mm. hakemistopalvelut. Nyt H.323-gatekeeperiä sinällään on hyvin vähän saatavilla kaupallisesti: esim. Ericssonin DataBeam on yksi esimerkki ja Customer-Connectista tehdään sellaista. On vähän harkinnanvaraista, voidaanko joitakin olemassaolevia hakemistopalveluita oikein konfiguroituina pitää gatekeepereinä. Tällaisia on esimerkiksi MS:n ILS-palvelin, jolle voidaan määrätä esimerkiksi siihen loggautumaan pystyvien IP-osoitteiden rajat (alueeseen pääsyn kontrollointi), henkilön nimi tämän IP-

osoitteen sijasta (erään asteen osoitteenmuunnos) ja yhteinen foorumi kaikille siihen sisäänloggautuneille (alueen hallinta). Ainoa, mihin se tiettävästi ei pysty, on kaistanleveyden kontrollointi.

Gatekeeperin tehtävistä kaistanleveyden sääteleminen on turhaa: se eroaa olennaisesti sen muista tehtävistä abstraktiotasonsa puolesta: alhaisen tason kaistanleveyden kontrolloinnin pitäisi olla alemman tason protokollien tehtävänä. Gatekeeperin pitäisi pystyä hallitsemaan käytössä olevan kaistanleveyden muutoksia ainoastaan käynnissä olevia puheluita rajoittamalla. CustomerConnect ei tule tähän gatekeeperin standardoituun toiminnallisuuteen puuttumaan.

3.1.3 Gatekeeper järjestelmässä

H.323:ssa gatekeeper näyttelee tärkeätä osaa. Se säätelee alueessa olevia resursseja, eli käytännössä säätelee pääsyä alueeseen sekä tarjoaa osoitteenmuunnospalveluita. Se toisin sanoen toimii sekä portinvartijana, että puhelun reitittimenä. Varsinkin ensinmainitusta tehtävästä johtuen H.323 määrittelee hyvin yksityiskohtaisesti eri menettelytavat käytettävissä olevien gatekeeperien määrästä riippuen.

Periaatteena gatekeeperin kanssa on se, että jos terminaalit sitä haluavat käyttää (esimerkiksi saadakseen pääsyluvan alueeseen tai ohjauspalveluita), ne rekisteröityvät siihen. Tämän jälkeen gatekeeper voi joko tarjota pelkän alustavan ohjauksen ja jättää lopun liikennöinnin terminaalien huoleksi, tai se voi myös päättää välittää kaikki pyynnöt keskitetysti, jolloin sen mahdollisuudet monitoroida terminaalien välillä kulkevia viestejä ja vastaavasti hallintamahdollisuudet paranevat huomattavasti.

Gatekeepereitä ei ole mitenkään rajoitettu olemaan vain yksi konferenssiin osallistuvien osapuolien käytössä. Standardi määrittelee myös useampien gatekeeperien keskinäisen signaloinnin tapauksissa, joissa osallistuvat terminaalit ovat rekisteröityneet eri gatekeepereihin. Terminaalin oletetaan rekisteröityvän puhelua kohden vain yhteen gatekeeperiin; jos terminaali lähettää GRQ:n (Gatekeeper Request, RAS-kanavassa [ks. Liite B]) broadcastina, vain yksi gatekeeper vastaa siihen myöntävästi.

3.2 ECMA:n CSTA-malli

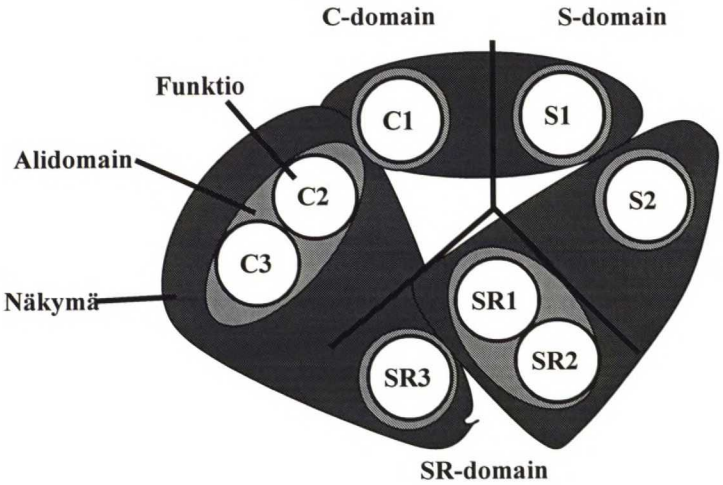
3.2.1 Yleistä

CSTA on lyhenne sanoista Computer Supported Telephony Applications, eli se pyrkii määrittelemään yleisiä sopimuksia tietokone-ohjatulle puheluidenkäsittelylle erityisesti PSTN-verkossa PBX-ympäristöön. Standardi on tietokoneteollisuuden European Computer Manufacturers' Associationin (ECMAn) määrittelemä tarkoituksena tarjota OSI-pinon ylimmän tason (OSI-level 7, Application Layer [STAL94]) palvelurajapinta vaihde-

komponenttien ja tietojenkäsittelykomponenttien välille [ECMA94]. CSTA on ennemminkin puhelunkäsittely*malli*, sillä se ei sisällä ohjelmointirajapintoja, eikä ota kantaa käytettyihin laitteistoihin, käyttöliittymiin tai verkkoprotokolliin. CSTA on siis hyvin yleisluontoinen standardi, ja se soveltuu melkein mihin tahansa puhelun-ohjaussovellukseen, joka pystyy tarjoamaan mallissa määritellyt palvelut ja tapahtumaraportit (event reports). Esimerkkinä CSTA:n ulkopuolelle jäävistä puhelin-järjestelmäkomponenteista ovat eräät (samassa CSTA-alueessa olevat) faksit ja modeemit, jotka eivät kykene selviytymään yhtäaikaa saapuvien carrier-signaalien aiheuttamista ristiriitatilanteista [ECMA94].

3.2.2 Arkkitehtuurista

CSTA on palvelurajapinta kahden eri funktioita suorittavan domainin välillä, jotka ovat tietojenkäsittely-domain (Computing Domain, C) ja vaihde-domain (Switching Domain, S). Standardi käsittelee molempien domainien tarvitsemia lisäpalveluita ja niitä tarjoavia komponentteja yhtenä tarkemmin määrittelemättömänä erikoisdomainina (Special Resource Domain, SR). Domainit on esitetty kuvassa 3.4.



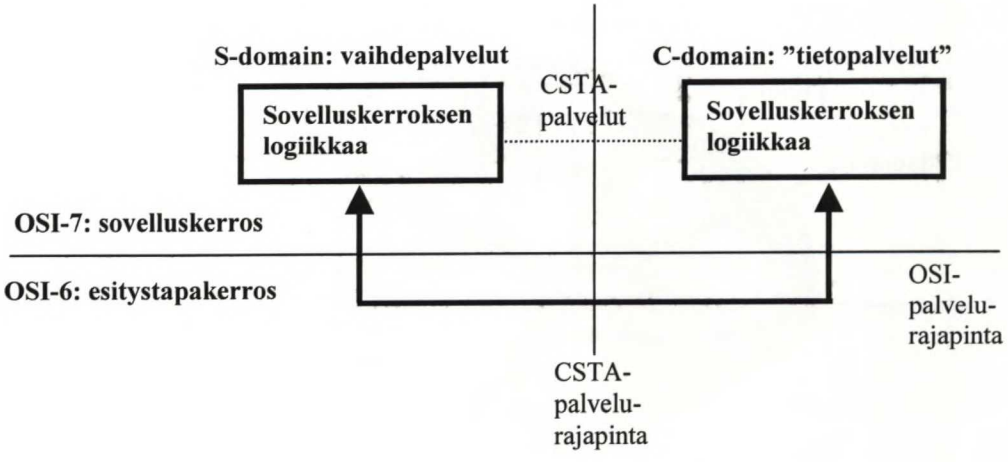
Kuva 3.4: CSTA:n domain-malli

Kuvassa valkoisella merkityt pienimmät komponentit ovat kunkin domainin toiminnallisuudelle määriteltyjä funktioita, kun taas sellaisten eri domainin kannalta yhtenevien funktioiden joukkoa, jotka ovat samassa domainissa (merkitty harmaalla) kutsutaan alidomainiksi. Itse CSTA-sovellus käsittää kaikki näkymät (merkitty tummanharmaalla) jotka eri domaineihin on määritelty.

C sisältää telekommunikaatiojärjestelmän ne tietokoneet ja sovellukset, joita voidaan kutsua suoraan S:stä. Nämä sovellukset toteuttavat ns. tietojenkäsittelyfunktion (Computing Function), joka on CSTA-mallin käsittelemän rajapinnan toisena osapuolena. Toisena osa-

puolena on nk. vaihdefunktio (Switching Function), joka on osa S:ää. S taas on niiden fyysisten vaihteiden ja näiden mahdollisten rajapintojen joukko, joiden palveluita voidaan pyytää suoraan C:stä käsin. SR:n muodostavat esimerkiksi datan säilyttämiseksi olevat tietokannat ja mahdolliset alemman tason signaalinkäsittely-funktiot, kuten kaistanleveyttä varaavat toiminnot. CustomerConnectissa S:n muodostaa yksi PBX, (esim. Siemensin Hicom) ja CustomerConnect-serverin vaihdeserveri. Vastaavasti C:n muodostavat tila-serveri ja osa clientin toiminnallisuudesta yhdessä. CustomerConnect-clientit, tietokanta ja VRU-yksikkö kuuluvat enimmäkseen SR:ään, joskaan rajat eivät ole suoraan komponenttien välisiä.

CSTA:n määrittelemät komponentit ovat client-server suhteessa keskenään: palvelua pyytävä komponentti on asiakkaan asemassa, ja palvelua tarjoava komponentti palvelimen asemassa. Koska sekä domainit S että C tyypillisesti tarvitsevat toistensa palveluja, on molempien domainien funktioiden pystyttävä toteuttamaan sekä client- että server-funktiot. Käytännön toteutus ei siis voi nojata pelkkiin synkronisiin kanaviin, vaan informaatio-kanavan on oltava koko ajan auki, ja molempien osapuolien on pystyttävä sekä lähettämään että vastaanottamaan myös pyytämättömiä (unsolicited) viestejä.



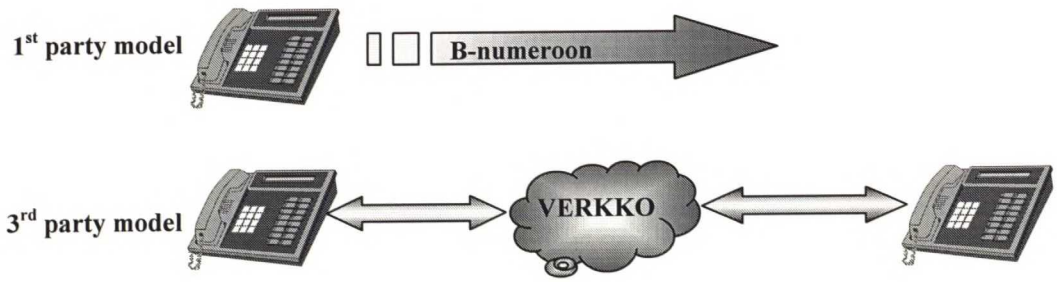
Kuva 3.5: CSTAn ja OSIn ero

OSI-pinon tarjoamiin rajapintoihin verrattuna CSTA-rajapinta on eri suunnassa kuin OSI-palvelurajapinta [ECMA94]. OSI-pinossa rajapinta on samassa pinossa kahdella eri tasolla olevien sovellusten välillä, kun taas CSTA:ssa kysymys on aina kahden yhdenvertaisen sovelluksen (tarkemmin ottaen kahden OSI-7 tason sovelluksen) keskinäisestä kommunikaatiosta.(Kuva 3.5)

CSTA:n rajoittuminen OSI-7:ään jättää ohjelmoijan harteille alempien tasojen, kuten viestitason ja protokollatason implementoimisen.

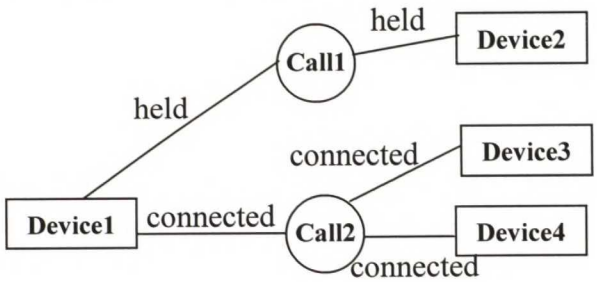
3.2.3 Objektimalli

CustomerConnect 1.5:ssä olennaisimpana osana CSTA:ta ovat S-domainin tarjoamat palvelut ja komponentit. Ne käydään tässä lyhyesti lävitse. CSTA:n puhelumalli on ns. "kolmannen osapuolen" malli (third-party model), joka erotuksena first-party mallista (esim. H.323, TAPI) käsittelee puhelua ulkopuolisen tarkkailijan roolista, jolloin puhelu koostuu kahdesta tai useammasta laitteesta, joiden välillä ääni (mediavirta, jos kyseessä olisi esim. videoneuvotteluvälineistö) kulkee. First-party mallit eivät ota huomioon muita laitteita kuin puhelusovelluksen käyttäjän hallinnassa olevan, jolloin näkökulma on toisen tilaajan / soittajan näkökanta. (Kuva 3.6)



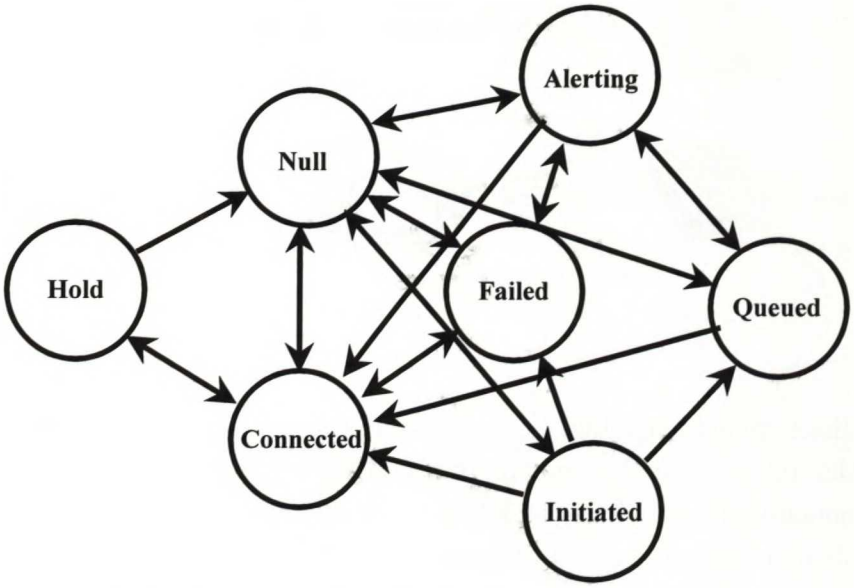
Kuva 3.6: 1st party vs. 3rd party model

Third-party mallissa puhelutapahtumaan osallistuu kolmen tyyppisiä objekteja: laitteet (device), yhteydet (connection) ja puhelu (call). Yhdessä puhelutapahtumassa voi olla useita laitteita, puheluita ja yhteyksiä yhtäaikaa. Laite vastaa puhelinta tai vastaavaa analogista tai digitaalista muistipaikkaa vaihteessa. Laitteella voi olla myös muita tyyppejä, kunhan se pystyy toteuttamaan mallissa määritellyt palvelut. Puhelu on yksi mediavirta. Virralla voi olla useampia lähteitä ja useampia määränpäitä, mutta kaikki puheluun kytkeytyneet laitteet voivat havainnoida samaa mediavirtaa. Yhteys on kuvaus laitteiden joukosta puheluiden joukkoon käänteiskuvauksineen. Kuvaus ei ole yksikäsitteinen, so. yksi laite voi olla kytkeytyneenä useampaan puheluun ja päinvastoin. Esimerkki puhelutapahtumasta esitettynä CSTA-mallin S-domainin objektein on kuvassa 3.7.



Kuva 3.7: Konferenssipuhelu, jossa Device1:tä käyttävä agentti on tekemässä konsultointipuhelua Device2:ta käyttävälle agentille.

Kuvassa 3.7 yhteyksien yläpuolelle on kirjoitettu niiden tilat. Kaikilla S-domainin objekteilla on tietty tila, jonka määrää edellinen tila ja toteutettu palvelupyyntö. Lisäksi jokaisella objektilla on oma yksikäsitteinen tunnisteensa, joka laitteilla on yleensä tämän puhelinnumero joko kanonisessa (universaalissa) tai alaliittymäformaatissa. Puheluilla tunniste on yleensä ainoastaan tarkasteludomainin alla yksikäsitteinen kokonaisluku, kun taas yhteyksillä tunniste on määritelty olemaan puhelu-ID:n ja laite-ID:n muodostama järjestetty pari (tosin käytännössä yhteydelläkin on oma tunniste-kokonaislukunsa ja lisäksi pointterit siihen assosioituihin puheluun ja laitteeseen.) Koska puhelun ja laitteiden tilat ovat niihin yhdistettyjen yhteysjoukon tiloista koottuja joukkoja, esitetään tässä suunnattuna graafina ainoastaan yhteysobjektin mahdolliset tilat ja tilansiirrot (kuva 3.8).



Kuva 3.8: yhteyksien mahdolliset tilat ja tilansiirrot niiden välillä.

Laitteen tila on siitä lähtevien yhteyksien tilojen joukko."Yksinkertainen" puhelun tila (Simple Call State) taas on paikalliseen laitteeseen menevän yhteyden ja "ensimmäisen" ulkopuoliseen laitteseen kulkevan yhteyden pari, jotka on nimetty erikseen. Yksinkertaisissa tapauksissa tämä Simple Call State on aivan riittävä, mutta on tilanteita (esim. konferenssipuhelu) joissa ei pärjätä pelkällä SCS:llä, vaan on turvauduttava laitetilan kaltaiseen yhdistettyyn puhelutilaan (Compound Call State). S-domainin objekteihin kuuluu vielä nk. agentti, joka esittää puhelinlaitteen ääressä istuvaa ihmistä. (Tarkemmin sanoen agentti-objekti esittää ACD-agentin toimintoja ACD-laitteen näkökulmasta) Kuten muillakin S-domainin objekteilla, agentillakin on tila ja tunniste. Tunniste on tyypillisesti kokonaisluku ja tila standardin mukaan yksi taulukossa 3.1 esitetyistä tiloista.

Taulukko 3.1: agentin tilat

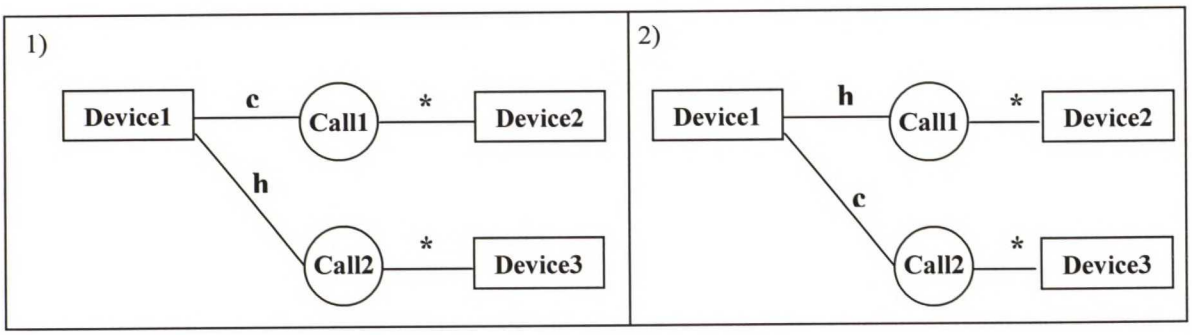
Agentin tila	Tilan selvitys
AGENT_NULL	Agentti ei ole systeemin kannalta läsnä.
AGENT_NOT_READY	Agentti läsnä, mutta ei kykene ottamaan vastaan ACD-puheluita
AGENT_READY	Agentti voi vastaanottaa mitä tahansa puheluita
AGENT_BUSY	Agentin laitteesta lähtee yhteys (tila != NULL) puheluun
AGENT_WORKING_ AFTER_CALL	Agentin "rauhoitusaika" puhelun jälkeen.

Ylläolevan taulukon tiloille ei tosin kaikille ole käyttöä CC 1.5:ssä, mutta toisaalta se ei myöskään ole riittävä, so. lisää tiloja on ollut pakko keksiä [HM&V98].

3.2.4 Palvelumäärittelyt

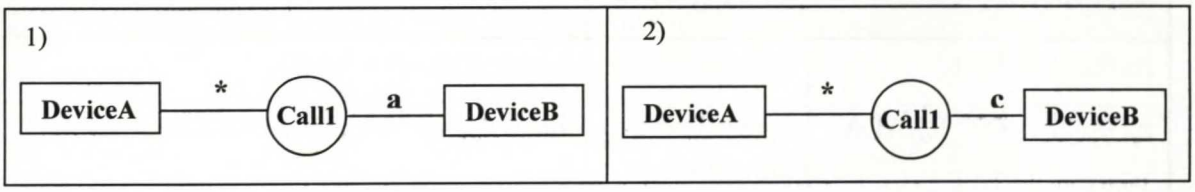
S-domainin tarjoamat palvelut, joita CC 1.5:ssä käytetään on esitetty seuraavassa. Yhteyksissä käytetyt merkinnät ovat: a = alerting, h = held, c = connected ja * = mikä tahansa tila.

1) **Alternate Call**. Tämä palvelu tarkoittaa konsultointipuhelussa mediavirran siirtämistä osapuolien A-B väliltä osapuolien A-C välille tai päinvastoin, riippuen mediavirran senhetkisestä kulkuvirrasta.

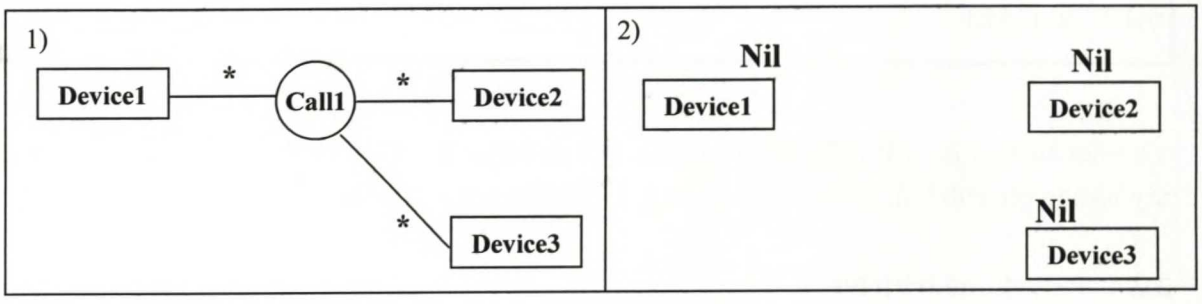


Tässä palvelussa toinen paikallinen yhteys laitetaan pitoon ja toinen otetaan pois pidosta. Etäyhteyden tilalla ei ole merkitystä [ECMA94].

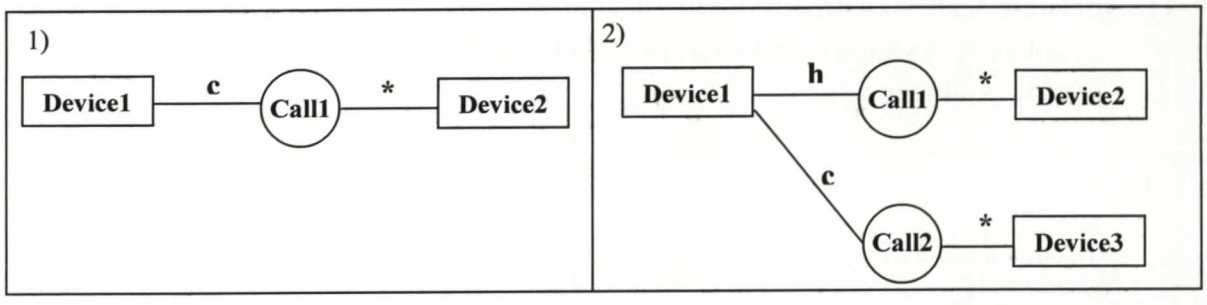
2) Answer Call. Tässä vastataan puheluun, l. siirretään paikallinen yhteys hälyttävästä tilasta yhteydelliseen tilaan. Etäyhteyden tilalla ei ole merkitystä [ECMA94].



3) Clear Call, l. puhelun lopetus. Tässä poistetaan kaikki puheluun liittyvät yhteydet niiden tilasta riippumatta, ja lopuksi itse puhelu [ECMA94].

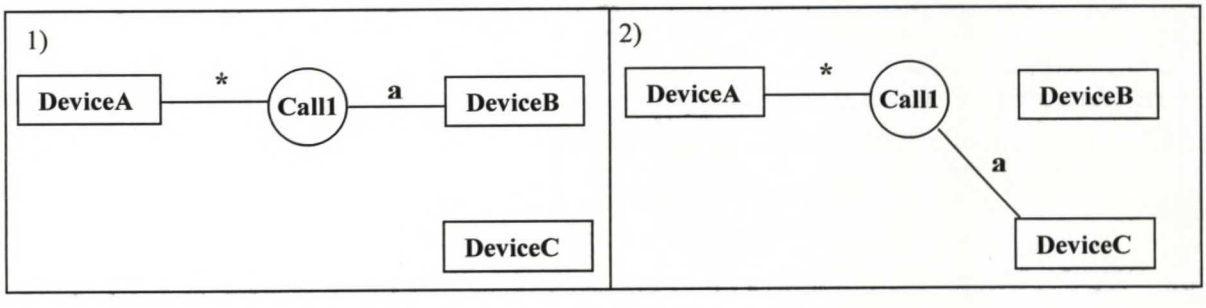


4) Consultation Call. Tämän tarkoituksena on kysyä joltakin toiselta agentilta neuvoa, samalla kun konsultoitava osapuoli odottaa linjalla.



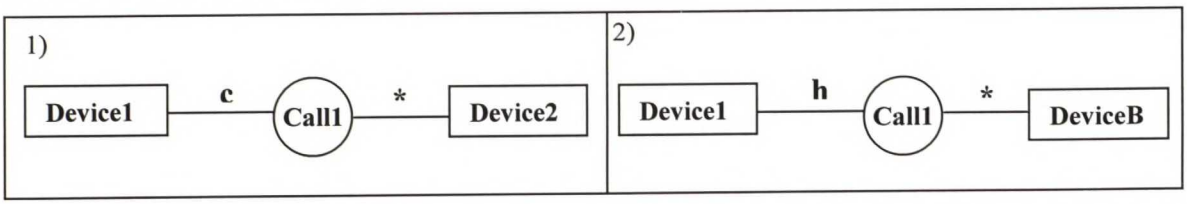
Tässä luodaan uusi puhelu, ja lopputuloksena paikallinen yhteys uuteen puheluun on yhteydellisessä tilassa, kun taas paikallinen yhteys vanhaan puheluun laitetaan pitoon. CSTA ei ota kantaa kumpaankaan etäyhteyteen [ECMA94].

5) Divert Call l. kutsunsiirto. Puhelu siirretään hälyttämästä yhdestä paikasta toiseen.



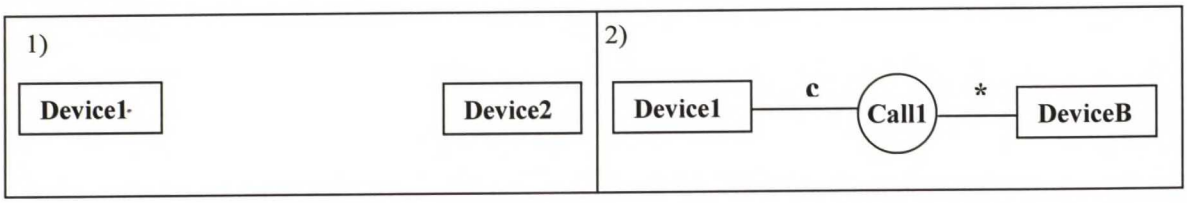
Vanha paikallinen yhteys poistetaan ja uusi luodaan hälyttävään tilaan eri laitteeseen. Etäyhteyteen (A-tilaajan laitteeseen) CSTA ei ota kantaa.

6) Hold Call l. puhelun pito on äänivirran pysäyttämistä ennen toista puhelinlaitetta yhteyden kuitenkaan katoamatta.



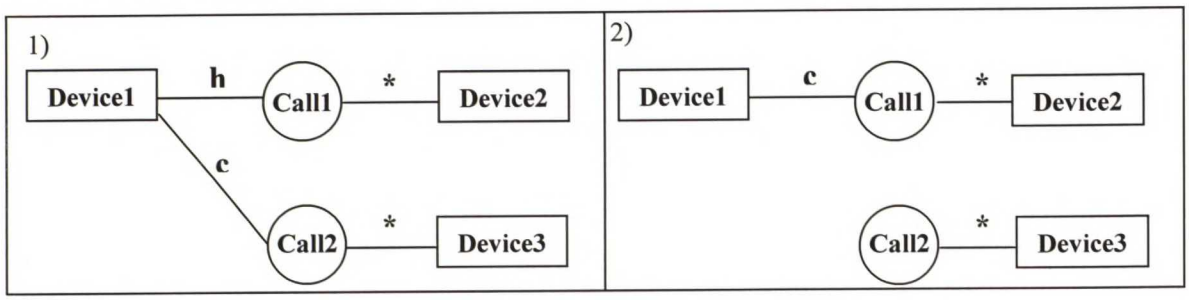
Olennaista on, että paikallinen yhteys ennen pitoon laittamista on tilassa "yhdistetty". Pitoon laitto on CSTA-mallin mukaisesti paikallisen yhteyden tilan muuttaminen yhdistetystä pidetyksi. (connected → held) [ECMA94].

7) Make Call, l. puhelun luonti paikallisesta päästä. Luodaan puhelu, ja yhteydet kahteen laitteeseen.



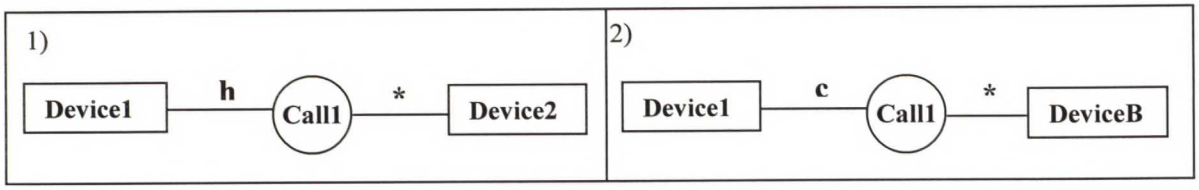
Paikallisen yhteyden tila tämän operaation jälkeen tulee olla yhdistetty. Etäyhteyden tilaan ei oteta kantaa [ECMA94].

8) Reconnect Call, l. konsultointipuhelun päättäminen katkaisemalla konsultointiyhteys.



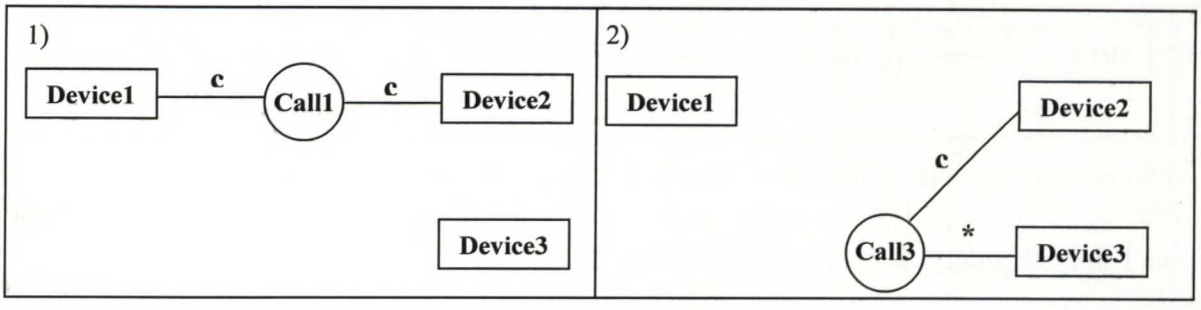
Tämä palvelu on kuten kohdassa 1, mutta uuden puhelun yhteys katkaistaan kokonaan [ECMA94].

9) Retrieve Call palauttaa puhelun pidosta, eli on kohdan 6 vastakohta. Aloittaa mediavirran lähettämisen uudelleen.



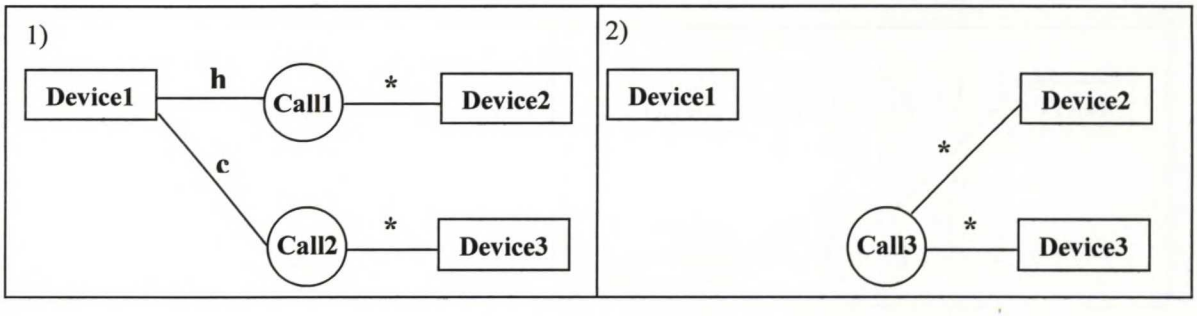
CSTA-malli määrittelee tämän palvelun yksinkertaisena paikallisen yhteyden tilanmuutoksena. Etäyhteyden tilaan ei oteta kantaa [ECMA94].

10) Single Step Transfer Call, eli puhelunsiirto siirtää mediavirran osapuolien A-B -väliltä pois siirtäjältä (A) osapuolien B-C välille. Erona konsultointipuheluun palvelun pyytäjä ei näe enää mediavirtaa ollenkaan, vaan tiputtautuu kokonaan pois puhelusta.



Alkuperäinen puhelu C1 lopetetaan yhteyksineen kokonaan ja uusi puhelu yhteyksineen luodaan osapuolien B ja C laitteiden välille. On huomattavata, että tilassa ennen palvelupyyntöä *sekä* paikallisen *että* etäyhteyden tilat ovat tunnettuja ja niiden oletetaan olevan "yhdistetty"-tiloissa [ECMA94]. Jos tämä ei ole mahdollista (esim Siemens Hicom, Ericsson MD110) [SIEM95] SSTransfer-palvelua ei voida toteuttaa. Nimi johtuu erosta välikyselylliseen siirtoon, kohta 11.

11) Transfer Call on puhelunsiirto kuten kohdassa 10, mutta siinä lähtötilanne on konsultointipuhelu. Tarkoituksena on, että jos neuvoa antava osapuoli tietääkin asiasta enemmän, puhelu voidaan siirtää suoraan näiden välille.



Kaksi paikallista tilaa ovat ennen palvelupyyntöä tiloissa "pidetty" ja "yhdistetty". Etäyhteyksien tiloilla ei ole väliä. Palvelupyynnön jälkeen molemmat vanhat puhelut on lopetettu, ja uusi puhelu yhteyksineen luotu uusien osapuolien välille. Molemmat uudet yhteydet ovat nyt palvelun alkuperäisen pyytäjän kannalta etäyhteyksiä, joten niiden tilaan ei oteta kantaa [ECMA94].

S-domainin palveluihin kuuluu myös tapahtumaviestin välittäminen C-domainille. Viestit jaetaan seitsemään luokkaan, joista tärkeimmät ovat agentin, puhelun ja VRU-yksikön tilanmuutoksista kertovat viestit. Mediavirran ohjauksen näkökulmasta ainoastaan puhelun ja agentin tilanmuutosviestit ovat lähemmän tarkastelun arvoisia. Agentin tilanmuutoksia

vastaa yksi viesti päätöstilaa kohden ja puhelutilojen kohdalla on joitakin palvelupyyntöjen CSTA-mallimuutoksia vastaamassa yksi tapahtumaviesti. Kaikkia CustomerConnectin palvelupyyntöjä kohden ei siis ole viestejä suoraan toimintoa vastaamassa.

On huomattava, että puhelun tilanmuutosviestit eivät ole varsinaisesti palvelupyyntöjen paluuarvoja tai niiden vahvistusviestejä, vaikka niitä siihenkin tarkoitukseen voidaan käyttää. S-domainin pitää nimittäin pystyä toimittamaan nämä viestit C-domainille riippumatta tilanmuutosten aiheuttajasta, joka voi aivan yhtä hyvin olla kokonaan S- ja C-domainien ulkopuolella kuin niiden sisäpuolellakin.

CSTA-mallissa C-domainille kuuluvat eräät korkeamman tason funktiot, kuten puhelun ohjaus päätepisteitä määrittelemällä ja puhelun reitin määrääminen. Tämän routing-funktion avulla voidaan hallita hyvin laaja-alaisesti inbound-puheluita, esimerkiksi ryhmäpuheluita, joka on tällä työllä aikaansaadun ohjauskonseptin testi-"case". Routing-funktiollakin voi tehdä puhelunohjausta, mutta se ei ole niiden päätarkoituksena. CustomerConnect ei käytä Routing-funktioita, vaan hoitaa puhelunohjauksen melko matalan tason menetelmillä, so. S-domainin palveluiden avulla. C-domain tekee toki paljoin muutakin kuin vain routingin CSTA-mallin mukaisista toiminnoista, mutta tässä esitetään vain ns. routing-toiminnot.

Routing-toiminnot ovat S-domainin pyytämiä palveluita C-domainilta. Koko routing-funktio on CSTA:ssa pilkottu viiteen alkeispalveluun, joita yhdistelemällä on mahdollista saada aikaan kaikki tällä konseptilla tarkoitetut funktiot. C-domain toteuttaa palvelut päättelemällä puhelun määränpään alustustietoihin tai tietokantaan määritellyistä parametreistä, jonotusrakenteista tai konsultoimalla ennaltamäärättyjä kriteerejä täyttäviä agentteja.

3.3 SCTP

SCTP, eli Simple Computer Telephony Protocol on alunperin Pacific Telephony Design-nimisen yrityksen luonnostelema mutta myöhemmin isojen telealan jättien (kuten Bell ja Nortel) täydentämänä versiona vuoden 1997 loppupuolella ilmestynyt avoin standardi client-server pohjaisille CTI-järjestelmille, erityisesti niiden ohjelmistokomponenteille [DAVI97].

SCTP on protokolla, joka tekstimuodossa välitettävine käskyineen pyrkii olemaan käyttö-järjestelmä- ja laitteistoriippumaton. Se on suunniteltu toimimaan kuten HTTP ja HTML – formaatti on hyvin lähellä HTML:ää – nimenomaan TCP/IP-protokollan kanssa, mikä käytännössä antaa sille koko Internetin soveltuvuusalueeksi. Suosituksen syntytapa muistuttaa SMTP:tä, eli SCTP:n tulisi olla dynaamisesti laajennettavissa, kuten muut RFC-ehdotukset. SCTP ei sisällä mitään binäärejä tai SDK:ta, ainoastaan spesifikaation mallista ja viesteistä.

Koska SCTP on verraten uusi suositus, ei sitä tukevia ohjelmistoja/järjestelmiä juuri ole, ja on mahdollista, että se jää muiden standardinomaisten rajapintojen jalkoihin turhana ylimääräisenä kerroksena, joten sitä ei esimerkkitapauksessa hyödynnetä. Sen sijaan käydään seuraavassa lävitse SCTP:n malli ja tärkeimpiä ominaisuuksia sen viesteistä pitäen silmällä USQ:n tulevaisuuden sovelluksia.

3.3.1 SCTP-malli

SCTP:n käyttämät objektit ovat H.323:n ja CSTA:n välimaastossa: näkökulma on kuten CSTA:ssa, mutta olettamukset ovat huomattavasti yleisempiä, pikemminkin kuten H.323:ssa. SCTP:n objekteja ovat: terminaalit, osoitteet, puhelut, istunnot, linjat ja ryhmät. Lisäksi termistöön kuuluvat mm. yhteydet (connection) ja viestit, joita käsitellään tarkemmin edempänä.

SCTP-malli olettaa ympäristökseen client-server-pohjaisen tietojenkäsittely-ympäristön, ts. yhden tai useamman palvelimen, joiden tarkoitus on olla valvonta- ja keskityskomponenttina jollekin järjestelmälle, sekä tyypillisesti palvelimia kertaluokkaa suuremman määrän clientteja, jotka toimivat lähinnä käyttöliittyminä ja yksinkertaisen logiikan laskentakomponentteina käyttäjälle päin. Analogia CustomerConnectiin merkitsisi yksi-yhteen kuvausta CustomerConnect-clientista SCTP-clienttiin ja vastaavasti servereihin. Ainoana erona on, että SCTP pyrkii olemaan sen verran joustava, että yhdellä clientilla voi olla useampi serveri samalla hetkellä, mihin CustomerConnectissa ei ole tarvetta. (H.323:ssa terminaali voi valita gatekeeperinsä, mutta kun se kerran on valittu, on gatekeeper yksikäsitteinen).

Malli pyrkii suureen toiminnallisuuteen yksinkertaisesti, joten se ei ole puhtaasti 1st party tai 3rd party – malli, vaan toiminnosta riippuen jompaa kumpaa.

Terminaalit tarkoittavat hyvin samanlaisia komponentteja kuin H.323:ssa, eli mitä tahansa SCTP-palvelimeen kytkettyjä (niin, että mahdollisella valvontaohjelmistolla on tieto kulloinkin suoritettavasta operaatiosta) laitteita tai sovelluksia, joiden kautta käyttäjä voi keskustella toisen käyttäjän kanssa. Ainakin toistaiseksi käyttäjien välisellä yhteydenpidolla viitataan reaaliaikaiseen yhteydenpitoon puhetta ja mahdollisesti videota hyväksikäyttäen kahden tai useamman osapuolen välillä. Terminaali on käyttäjäriippumaton, ts. terminaalin ja käyttäjän välinen assosiaatio on hyvin dynaaminen luonteeltaan. Terminaali tunnistetaan sen käyttämästä tunnistetekstistä, joka on tyypillisesti ASCII:na välitetty kokonaisluku [DAVI97].

Osoitteet ovat käyttäjien tunnistetietoja, joiden avulla jokainen käyttäjä voidaan identifioida yksikäsitteisesti halutulla alueella (oletuksena globaalisti). Osoitteen formaattiin ei oteta kantaa, kunhan se on ASCII:na koodattu ja välitetty [DAVI97]. Käyttäjä ei välttämättä ole osoitteen kannalta yksittäinen henkilö, koska SCTP ei ota kantaa sen varsinaiseen käyttöön. Kuvaus osoiteavaruudesta käyttäjäavaruuteen on useasta-useaan.

Esimerkiksi osoitteesta käy henkilön nimi, ryhmän nimi tai sen joku alias samoin kuin sähköpostiosoite, tms.

Puhelun SCTP määrittelee niiksi informaatiovirroiksi, jotka kulkevat yhden tai useamman osapuolen välillä siten, että kaikki samaan virtaan kytkeytyneet havaitsevat samanlaisen informaatiovirran (ts. ne pisteestä pisteeseen olevat kommunikointiyhteydet, jotka kaikissa päätepisteissä havaitaan samanlaisina, eli tavalliset tai konferenssipuhelut). Assosioinnit puhelun ja terminaalien, samoin kuin puheluiden ja osoitteiden välillä eivät ole toisensa poissulkevia, eli kuvaukset kustakin joukosta toiseen ovat yleisiä useasta-useaan – kuvauksia [DAVI97].

Istunnot ovat SCTP-clienttien datayhteyksiä SCTP-serveriin, esimerkiksi terminaalin (client) käymät viestit valvontaohjelmiston (serveri) kanssa ensimmäisestä auktorisoinnista viimeiseen lopetusviestiin.

Linjat on lisätty SCTP:hen PBX-tuen vuoksi, ts. mahdollistamaan ulkoisten vaihdelinjojen ja vaihteesta ulospäin menevien linjojen monitoroinnin. Linjat viittaavat PBX:n tai vastaavan audiota prosessoivan komponentin audiota liikutteleviin piireihin sekä näiden kaltaisesti toimiviin virtuaalilinjoihin [DAVI97].Tämä kompromissi absraktiotasoon perinteisten vaihteiden eduksi vähentää osaltaan SCTP:n arvoa mediariippumattoman järjestelmän protokollana, mutta lisää luonnollisesti sen sopivuuttaa isoon osaan olemassaolevista järjestelmistä.

Useiden osoitteiden loogisia kokonaisuuksia voidaan määritellä **ryhmi**ksi esim. ACD-toiminnallisuutta varten. Tätä ei tosin ole vielä toteutettu protokollaan, ja olisi osittain osoitemäärittelyn päällä (yksi osoite voi käsittää useita terminaaleja / käyttäjiä). (Koska viestit toimivat osoitteiden perusteella, ei niitä voi sellaisenaan käyttää ACD:hen, vaan tarvittaisiin jokin osoitekokoelman objekti, jota ei toistaiseksi ole – ei ole tosin ACD:n viestejäkään)

Viestit, jotka liikkuvat SCTP-järjestelmässä, kulkevat **yhteyksiä** pitkin. Yhteydet voidaan synnyttää clientin ja serverin välille. Clienttien keskinäiseen viestinvaihtoon samoin kuin serveriltä serverille kulkeviin viestiin SCTP ei ota kantaa [DAVI97], ja kaikki viestit onkin määritelty kulkevaksi nimenomaan clientilta serverille tai päinvastoin. Yhteydet toimivat sockettien päällä, joiden portti on tarkoin määritelty. Sockettien oletetaan olevan yhteydellisiä, ts. informaatiovirtaa ei tarvitse avata joka kerran uudelleen, kun lähetetään uusi viesti.

Viestien formaatti on muokattu lähes samanlaiseksi WWW-selaimen POST-metodilla lähetettyjen pyyntöjen formaatin kanssa. Tämä mahdollistaa useiden valmisparsereiden yms. käytön, ja periaatteessa tavallinen pelkästään HTML:n varassa toimiva selain voi toimia SCTP-clienttina.

3.3.2 SCTP-viesteistä

Kuten edellä on kerrottu, SCTP-viestit ovat alemmista 126:sta ASCII-merkistä koostuvia tekstirivejä, jotka aina päättyvät rivinvaihtoon (so. CR+LF). Viestien parametrit ja niiden arvot erotellaan kuten CGI-kutsussa, mutta vastauksen ei tarvitse tulla synkronisesti, kuten CGI-kutsussa oletetaan [DAVI97].

SCTP-viestit jaetaan kolmeen tyyppiin: istunnon synnyttämiseksi, ylläpitämiseksi ja lopettamiseksi tarkoitetut viestit; transaktiot ja lopuksi asynkroniset tapahtumatyyppiset viestit. Ensinmainitut istuntoviestit ovat luonteeltaan sekä transaktioita että asynkronisia tapahtumia, ja ne sisältävät sellaisia viestejä, kuten *login*, *logout* ja *heartbeat*, jolla serveri voi tarkistaa clientin olemassaolon. Transaktioissa viestit ovat aina jonkin asteen palvelupyyntöjä tai komentoja, joihin odotetaan aina vastausta. Transaktioilla on pakollisena parametrina nk. transaktio-ID, eli TID, joiden avulla vastaukset on mahdollista yhdistää oikeisiin pyyntöihin. Asynkroniset tapahtumat, kuten hälyttävä puhelu, ovat viestejä, joita toinen osapuoli ei ole mitenkään erikoisesti pyytänyt (paitsi tietyntyyppisten tapahtumien välityspyyntöä), eikä niihin odoteta vastausta.

Puhelunohjausviesteissä kommunikointi tapahtuu vähintään puhelu-ID:n (CID) ja TID:n avulla. Näiden tunnisteiden luontimekanismiin tai formaattiin protokolla ei ota kantaa, kunhan ne ovat yksikäsitteisiä järjestelmässä. (Tämä saattaa tuoda muassaan järjestelmien välisen kommunikaation mahdottomuuden, ts. voi olla työlästä luoda nimeämissysteemi, missä useampien vuorovaikuttavien SCTP-järjestelmien viestien parametrit ovat yksikäsitteisiä.)

Transaktioista tärkeimmät lienevät puhelunohjausviestit, joskin osoitteenkäsittelyviestit tarjoavat nekin käyttäjän kannalta tärkeitä palveluita. Puhelunohjauksen osalta tuetaan kaikkia tärkeimpiä yksinkertaisia puheluohjaustoimintoja, sekä konferenssipuheluja. Käytännössä esimerkiksi kaikkia CustomerConnectin tukemia palveluja SCTP:llä ei voi toteuttaa (puuttumaan jäävät esimerkiksi jonotus ja soittolistat ym. ACD-toiminnallisuus), mutta pienen toimiston tarpeet SCTP:llä kyllä voidaan tyydyttää.

3.4 NetMeeting-API – rajapinta

NetMeeting 2.1 on Micorsoftin palvelimilta vapaasti ladattavissa oleva IP-puheluohjelmisto [MICR97b]. NetMeeting tukee H.323-standardia muuten, paitsi gatekeeperin osalta. Tämä puutteen korjaaminen on yksi tämän työn keskeisistä toimenpiteistä.

Rajapinnasta käytetään/käytettäisiin taulukossa 3.2 (alla) nimettyjä konferensointifunktioita.

Taulukko 3.2: NetMeeting-APIn konferensointifunktiot

Funktio	Tuettu (NM 2.1 API)	Merkitys
IConferenceX::Invite	On	Kutsutaan jo luotuun konferenssiin uusi jäsen. Jos jäsen on järjestyksessä toinen liittynyt, synnyttää NMmediavirran l. aloittaa puhelun. Toist. ainoa keino aloittaa kahdenvälinenkin puhelu (outboundina)
IConferenceX::Accept/ RejectInvite	Ei	Hyväksytään/hylätään jonkin tahon kutsu liittyä konferenssiin. Näiden avulla voitaisiin kontrolloida inbound-puheluita käyttämättä virtuaalisia soitonkäsitteitä.
IConferenceX::Join	On	Ulkopuolinen jäsen haluaa liittyä konferenssiin: vastaa inbound-puhelua. Ainoa keino Inviten lisäksi aloittaa kahdenvälinen puhelu.
IConferenceX::Accept/RejectJoin	Ei	Hyväksytään/hylätään jonkin tahon pyyntö liittyä konferenssiin. Näitäkin voitaisiin käyttää inbound-puhelujen kontrollointiin ilman virtuaalipuheluja.
IConferenceX::CreateChannel	Ei	Kanavanluonti; tarvittaisiin esim. puhelunsiirtoon ja inbound-ohjaukseen ilman virtuaalipuhelukäsitettä.
IConferenceX::Leave	On	Poistutaan konferenssista. Jos lähtijä on isäntä, kaikki siihen isäntäsuhteessa "alapuolella" olevat pakotetaan myös jättämään konferenssi. Kahdenvälisessä puhelussa yhteys katkeaa riippumatta siitä, kuka "Hangup"-nappia painaa.
IConfChannelX::Include/Exclude Member	Ei	Toinen tapa kanavanluonnin rinnalla aikaansaada puhelunsiirto.

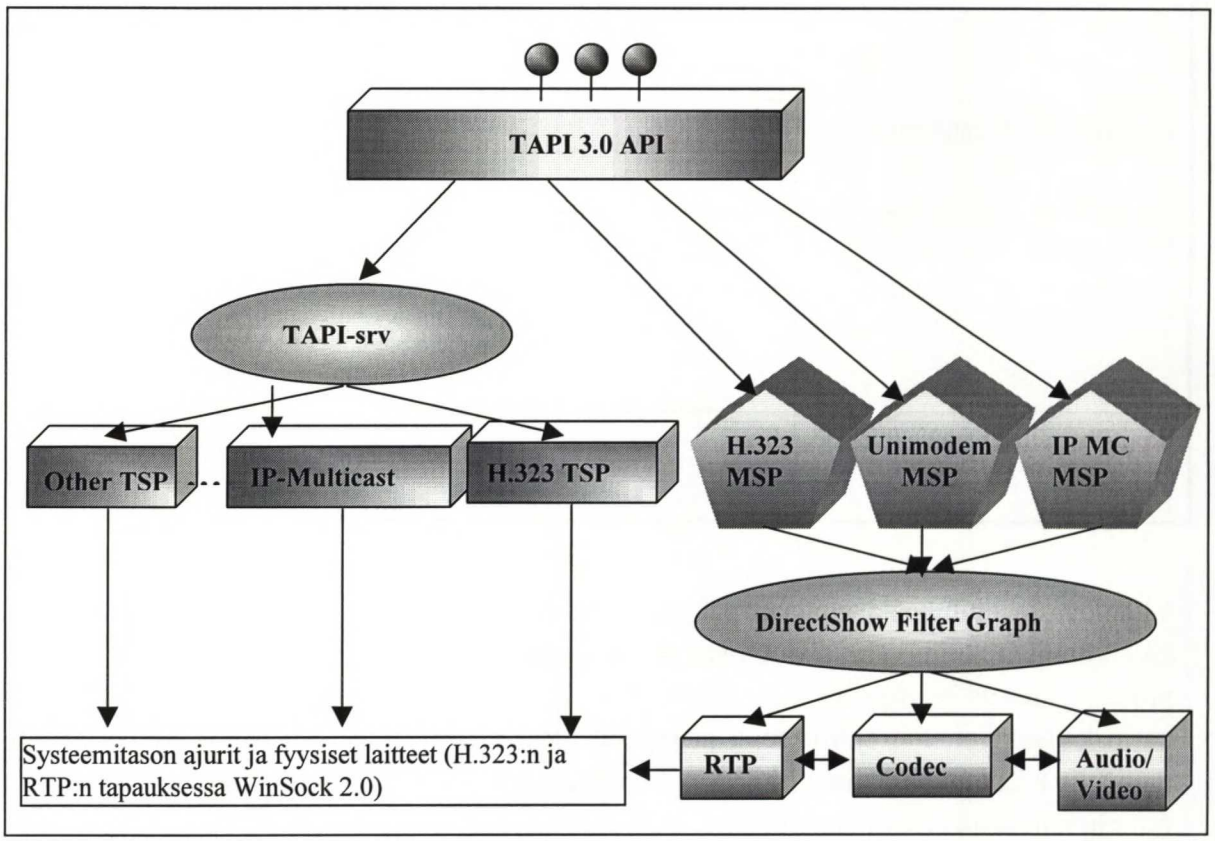
Ylläolevasta taulukosta voidaan päätellä, että NetMeeting 2.0-ActiveX-kontrollin kanssa eivät onnistu kuin vain aivan perustavaa laatua olevat puhelunohjaustoimenpiteet, eli puhelun aloitus ja lopetus. (Toisaalta virtuaalisessa puhelunohjausympäristössä gatekeeperin kanssa puhelun aloituksella ja lopetuksellakin saavutetaan jo hyväksyttävä toiminnallisuus.) Taulukossa mainittujen Accept/Reject-funktioiden tuen puute johtuu ennemminkin siitä, että NetMeeting pitää vastaavat tapahtumat (event) itsellään, kuin niiden varsinaisista implementoinnin vajeista. (Virtuaalisiin puheluihin turvaudutaan MP:n puuttumisen lisäksi siksi, että NetMeeting ei tarjoa reaalisista puheluista tarpeeksi informaatiota.)

3.5 TAPI 3.0 – rajapinta

TAPI, l. Telephony API, on Microsoftin lanseeraama ohjelmointirajapinta käytettäväksi lähinnä Windows-systeemeistä käsin reaaliaikaisen viestinnän komponenttien ohjaamiseksi ohjelmallisesti [MICR97c]. Sen seuraava isompaa versiota, 3.0 on lupailtu ilmestyväksi NT 2000:n mukana. Valitettavasti tätä kirjoitettaessa TAPI 3:sta ei ole saatavissa White-Paperiä yksityiskohtaisempia dokumentteja.

TAPI 3.0 eroaa edeltäjistään melko radikaalisti paitsi ohjelmointimallinsa, myös yleisyytensä puolesta: sen tarkoituksena on tarjota yhtenäinen ja sovellusohjelmoijalle läpinäkymätön rajapinta kaikentyyppisten mediastreamien ohjausta varten – streamin fyysisestä sijainnista riippumatta [MICR97c].

TAPI 3.0 sisältää peruspaketissaan H.323-tuen (eli standardia tukevan MSP-ajurin), joten se tulee olemaan ideaaliratkaisu Windows-pohjaisille CTI-järjestelmille. Osa TAPI 3.0:n järjestelmäarkkitehtuurista on esitetty kuvassa 3.9:



Kuva 3.9: Osa TAPI 3.0:n arkkitehtuurista

TAPI 3.0:n objekteihin tai ohjelmointimalliin ei tässä puututa. Sen sijaan käydään yleisen toimintamallin lisäksi lävitse mediavirtamalli, sekä tärkeimpinä uutuuksina palvelunlaadusta ja salauksesta huolehtiminen.

3.5.1 Arkkitehtuurin toimintaperiaatteet

Kuten kuvasta 3.9 voidaan todeta, teleliikennepalveluntarjoajat (TSP) ja medivirtapalvelujen tarjoajat (MSP) ovat varsin keskeisessä asemassa TAPI 3.0:n toiminnassa. Aikaisempiin versioihin (esim. TAPI 2.1) verrattuna samaa on TSP-komponenttien käyttö puhelunohjaustoimintojen abstrahoinnissa. Uutta on sen sijaan MSP:den käyttö IP-verkon kautta kulkevien mediavirtojen ohjauksessa.

TSP:t on tarkoitettu puheluiden (ja yleistettyjen mediavirtojen puhelunomaisen käsittelyn) ohjaukseen mahdollisimman geneerisellä tavalla. Jos puhelun jokin osa tunnistetaan esimerkiksi PSTN-signaaliksi, käytetään kyseisen PBX:n tai vastaavan mukana tullutta ajuria tämän TSP:n kautta, eikä välttämättä käytetä ollenkaan MSP:tä hyväksi. Jos taas kysymyksessä on vaikkapa H.323-tuettu IP-puhelu, käytetään TAPI 3.0:n H.323-TSP:tä, joka toteuttaa standardissa määritellyn kontrollisignalointipinon, ja kutsuu sen jälkeen H.323-MSP:tä. H.323-MSP ohjaa mediavirrat toivotulla tavalla vuorostaan DirectShow API:a ja suodingraafia hyväksikäyttäen.

MSP:tä voi käyttää halutessaan myös suoraan, (Media Stream Control – objektien kautta) mutta se on vastaavasti yhtä abstraktiotasoa alempana [MICR97c]. MSP:t ja mediavirtamalli ovat TAPI 3.0:n ”työhevosia”, mitä tulee todelliseen mediariippumattomaan kommunikointiin. TSP:den kautta voidaan kyllä ohjata samantyyppisiä puheluita riippumattomasti, mutta erityyppisten puheluiden sekoittaminen saman puhelutapahtuman alla vaatii MSP:tä hyväkseen.

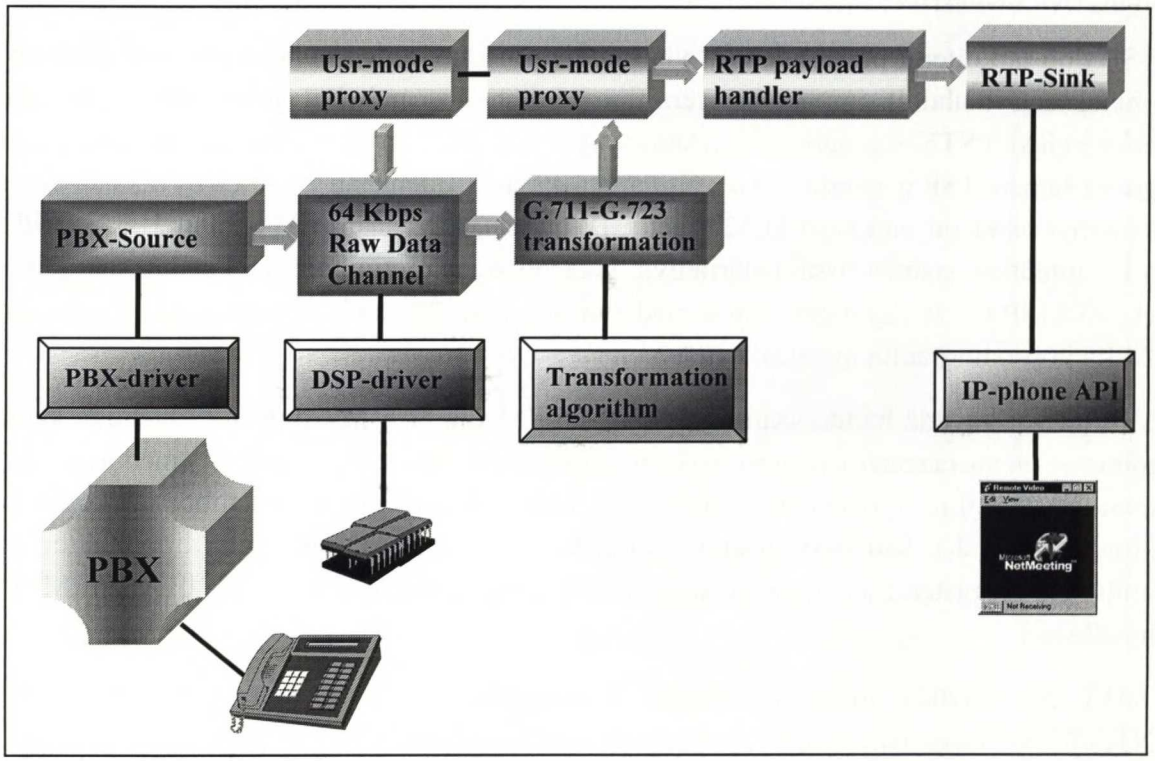
TAPI 3.0 käyttää myös hyväkseen (Microsoftin) eräitä hakemistopalveluja, kuten NT2000:n Active Directoryä, pystyäkseen mm. konferensoinnissa pysymään ajan tasalla osallistujien osoitteista, ja lisäksi tulevat yhteensopivuudet aiempien TAPI-versioiden kanssa, mutta kummatkaan eivät kuulu varsinaisesti tähän kuvaukseen.

3.5.2 Mediavirtamalli

Suurin osa mediariippumattomuudesta on upotettu TAPI:n nk. mediavirtamalliin. (Itse asiassa, jos mediariippumattomat osat tehdään jotenkin muuten kuin mediavirtamallin mukaisesti, TAPI 3.0 ei puutu siihen.) Mediavirtamalli pitää sisällään filttereiksi kutsutuista komponenteista koostuvan suunnatun verkon, filtterigraafin. Tämä graafi toimii mallin ytimenä, ja sitä käyttävät eri TSP:t kunkin filtterin objektirajapinnan kautta. Vastaavasti filtterit käyttävät tarpeen mukaan laiteajureita ja erilaisia koodausalgoritmeja (esim. PSTN-IP-yhdyskäytäväfiltteri voisi käyttää vaikkapa Dialogicin korttien laiteajureita ja G.711-G.723 – kooderia/muunnosohjelmaa).

Eri filtterit eivät ole mitenkään staattisesti sidottuja, vaan graafi kootaan tapauskohtaisesti jokaiselle puhelutapahtumalle erikseen. Filtterit voidaan tarvittaessa identifioida sitä konstruoivan TSP:n käyttämin käsittein: yksi voi näytellä yhdyskäytävän virkaa, yksi

terminaalin ja yksi MP:n sisältävän MCU:n roolia. Tärkeintä on, että kukin suodin määritelmän mukaisesti pystyy toimimaan mediavirran lähteenä ja/tai nieluna, tyypillisesti toisena vain, jos se esittää terminaalia tai vastaavaa loppukäyttäjälle näkyvää laitetta. Kuvassa 3.10 on esimerkki suodingraafista, joka syntyy, jos PSTN-puhelimesta soitetaan IP-puhelimeen.



Kuva 3.10: Esimerkki TAPI 3.0:n filtterigraafista

Kuvan 3.10 ylemmän rivin komponentit ovat tavallisen käyttäjän oikeuksilla toimivia, ja seuraavan rivin kernelin oikeuksilla ajavia prosesseja. Kyseiset oikeustasojen erot johtuvat tehokkuusnäkökohdista [MICR97c]. Toiseksi alimmalla rivillä on esitetty kuhunkin solmuun liittyvät ajurit tai muut ohjelmat/rajapinnat, joiden kautta solmu pystyy toteuttamaan tarkoituksensa.

CTI-sovelluksen keino hallita mediavirtoja TAPI 3.0:n kautta tapahtuu juuri em. filtterigraafia muokkaamalla: kun saapuva puhelu ja sen päätepiste ovat hallinnassa, voi MSP tarkistaa, onko järjestelmässä kyseisen streamin käsittelyyn sopivia filttereitä, ja konstruoida niistä sopivan graafin. Kun graafi on konstruoitu, ohjataan mediavirta sen lähteistä filtterien lävitse ja näiden käsitteleminä nieluina toimiville solmuille (terminaaleille).

3.5.3 TAPI 3.0:n muita lupauksia

TAPI 3.0 on (tulee olemaan) ensimmäisiä laaja-alaisia rajapintoja mediavirtojen yhtenäiseen käsittelyyn. Tämän yhtenäistämisen lisäksi TAPI 3.0:ssa otetaan huomioon myös toistaiseksi vähemmälle huomiolle jääneet seikat: palvelunlaatu ja tiedon salaus [MICR97c].

Palvelunlaatua voidaan parantaa siirtämällä mediavirta kokonaan SCN:ään, kuten eräät jo olemassaolevat sovellukset (lähinnä yhdyskäytävien valmistajien tekemät [EFUS97]) toimivat, tai käyttää Internetiin kehitettyjä mekanismeja viiveiden ja kaistanleveyden varmistamiseksi. TAPI 3.0 käyttää viimeksi mainittuja metodeja sisältäen Resource Reservation Protocol -RSVP:n kaistanleveyden varaamiseksi, sekä eräitä standardoituja useampiin OSI-pinoihin kuuluvia paketin prioriteettia nostavia keinoja, esim.802.1p:tä Data-Link – kerroksessa ja IP-palvelunlaatua (IP Type of Service) ylemmissä kerroksissa [MICR97c].

IP-verkossa liikkuva mediavirta on periaatteessa suojaamatonta, joten kuka tahansa voi kytkeytyä linjalle ja tehdä itsestään mediavirran yhden nielun. TAPI 3.0:aan sisältyy konferensseissa liikkuvan mediavirran kryptaus, jonka avaimienhallinta on melko kiinteästi kytketty NT2000:n turvamekanismeihin ja varsinkin Active Directoryyn. Kryptauksessa käytetään epäsymmetrisiä salausmenetelmiä [MICR97c].

3.6 Standardien ja rajapintojen vertailua

3.6.1 CSTA, H.323 ja SCTP

CSTA:n ja H.323:n perimmäinen ero on niiden käyttötarkoituksessa. CSTA tehtiin tavanomaisille PSTN-verkossa liikkuvalle äänivirralle, jota voidaan *kontrolloida* tietokoneen välityksellä tietoverkossa. H.323 sen sijaan on askelen edellä siinä, että sen määritelmien mukaan myös mediavirta kulkee samana alatason paketteina samaa fyysistä mediaa pitkin kuin itse ohjausdata [ITUT96].

CSTA on lisäksi teollisuuden kehittelemä korkean tason standardi, joka ottaa kyllä huomioon senhetkiset teollisuuden tarpeet, muttei pyri paneutumaan syvällisemmin esimerkiksi järjestelmän kehitysmahdollisuuksiin tai sen teoreettisiin perusteisiin. H.323:n rajapinnatkin ovat OSI-malliin nähden kohtisuorassa, l. ainoastaan saman OSI-tason komponenttien hyödynnettävissä, mutta rajapinnat ulottuvat paljon sovellustasoa alemmas, osin melkein protokollatasolle.

CSTA ei juuri välitä systeemin arkkitehtuurista muuten kuin client-server-pakotteisesti. H.323 määrittelee osallistuvat komponentit paljon yksityiskohtaisemmin kuin pelkällä domain-tasolla.

Ylläolevasta voisi päätellä, että H.323 on CSTA:n tarkennus ja laajennus, missä tapauksessa H.323-järjestelmän yhteensovittaminen CSTA-järjestelmään ei tuottaisi suuriakaan vaikeuksia. Molemmat järjestelmät määrittelevät kuitenkin sovellustasolla omia objektejaan ja viestejään, jotka ovat yhteensopivia ainoastaan pieneltä osin. Standardit ovat eri maailmoista ja kumpikaan ei tunne toistaan. H.323 kylläkin määrittelee "ulkoisia" järjestelmiä joihin on mahdollista kytkeytyä yhdyskäytävien avulla [ITUT96], ja CSTA:n SR-domain kuvaa periaatteessa CSTA:han sopimattomia järjestelmiä [ECMA94], mutta standardit käsittelevät samaa sovellusaluetta, joten kumpikaan ei ole toisen kannalta täysin ulkopuolinen. Vaikka suurin osa työstä kuluikin epäkypsien valmisohjelmien yhteensovittamiseen ja tästä aiheutuvan uudenlaisen vaihdeproblematiikan miettimiseen, kahta eri maailmasta olevaa standardia tottelevien järjestelmien yhteensovittaminen ei kuitenkaan ole työn pienin osa.

H.323:n käyttämä puhelumalli on erilainen CSTA:han nähden. H.323 käyttää TAPIn lailla 1st-party-mallia, vaikka sitä ei suoraan ilmaistakaan: kaikissa RAS- ja puhelunohjauskanavan viesteissä on tunnisteena nk. CRV, eli Call Reference Value [ITUT96], jonka avulla kaikki operaatiot tehdään. Lisäksi konferenssipuhelu ei ole mikä tahansa puhelu, vaan oma objektinsa johon puheluobjektin tulee liittyä. Molemmat ovat 1st-party-mallin tunnusmerkkejä: 3rd-party-mallissa pelkkä puhelun tunniste ei riitä, ja toisaalta konferenssi ei 3rd-party-mallissa ole muuta kuin puhelu, johon on kiinnittynyt enemmän kuin kaksi CONNECTED-tilassa olevaa yhteyttä.

CSTA:n ja H.323:n mukaan toimivien järjestelmien yhteensovittaminen vaatii siis myös kuvauksen 3rd-party-mallin ja 1st-party-mallin välillä, jossa tyypillisesti jokaista 3rd-party-mallin yhteyttä vastaa 1st-party-mallin puhelu, siinä missä laitteet ovat molemmissa malleissa yhteneviä. 3rd-party-mallin puheluobjektin kaltaista objektia ei 1st-party-mallissa konferenssiobjektia lukuunottamatta ole. Tällöin 1st-party-mallin mukaan toimivan ohjaussysteemin on pidettävä sisäisissä rakenteissaan tieto eri puheluiden välisistä yhteyksistä, mikä taas johtaa vahvasti 3rd-party-mallin puolelle. Tämän vuoksi ohjausjärjestelmän on jopa yksinkertaisempaa pitäytyä 3rd-party-mallissa, ja samalla antaa vaihdekomponenttien käyttää 1st-party-mallia, koska niillä ei ole mitään tarvetta monimutkaisempaan käsittelyyn.

H.323 kypsyy koko ajan, erityisesti viimeksi mukaan tulleet gatekeepereiden välisen kommunikaation spesifikaatiot tekevät suosituksesta varteenotettavamman. Eräs sen heikkoja kohtia on edelleen osoitteiden haku. Suositus "olettaa" eräiden kanavien osoitteiden olevan tunnettuja, muttei ota kantaa siihen, kuinka osoitteet on julkistettu. CSTA:han tämä argumentti ei päde, koska sen rajapinta on niin korkealla tasolla, ja tyypillisesti sen tarkoittamat komponentit sijaitsevat fyysisesti hyvin pienellä ja/tai hyvin tunnetulla alueella, missä osoitteistus on lähes itsestäänselvyys. Hajautetun ja dynaamisen järjestelmän osoitteistukseen CSTA ei ota kantaa.

SCTP on nimensä mukaisesti kompakti, yksinkertaiseen puhelunohjaukseen helpon rajapinnan kautta käytettäväksi tarkoitettu protokolla; yhteensopivan laitteiston ja SDK:n avulla SCTP:llä pystytään tekemään esimerkiksi pienen yrityksen puhelunohjausoperaatiot. SCTP:n vahvuus on nimenomaan Internet-puheluissa, ts. päätteen kautta kulkevassa ja sitä kautta ohjattavissa olevasta puhelinliikenteessä. Tämä johtuu ohjauslogiikan sijainnista: jos logiikka on standardissa tietokoneessa, se on helppoa tehdä sinne vaikkapa SCTP:llä. PBXien ja muun PSTN-maailman käyttämät ratkaisut ovat suhteellisen kankeita ja monimutkaisia SCTP:tä varten.

CSTA:han verrattuna SCTP on hyvin kevyt malli: SCTP keskittyy viesteihin, so. ohjausdatan informaatiovirtaan siinä, missä CSTA pyrkii syleilemään koko puhelumaailmaa mallillaan. Toisaalta vaativaan Call-Center – käyttöön SCTP ei sisällä tarpeeksi globaalisuutta, hajautuneisuutta ja toiminnallisuutta. Client-Server-pohjaisena SCTP:n käyttämä malli ei skaalaudu kovin hyvin tapauksiin, joissa käyttäjämäärä on tuhansia, ehkä jopa miljoonia. PSTN-maailman informaatiovirtoja käsittelevät laitteistot, kuten PBX:t, eivät usein salli tällaisia suuruusluokkia mediavirtojen lukumäärässä. Jos sen sijaan haetaan mediariippumatonta systeemiä, täytyy ottaa huomioon IP-puheluiden mahdollinen volyymi – periaatteessa koko Internet. SCTP sellaisenaan ei siis sovi mediariippumattomien puhelunohjausjärjestelmien toteutusmalliksi.

Mediariippumattomassa järjestelmässä on lisäksi otettava huomioon yhteensovittamisongelmat muiden standardien kanssa, kuten on asia H.323:n ja CSTA:n välillä. SCTP on soveltamisalueeltaan paljon lähempänä H.323:a kuin CSTA:ta. Näinollen SCTP:n yhteensovittaminen CSTA:n kanssa, mikä tehtävä käytännössä tulisi väistämättä eteen, kohtaisi samat vaikeudet kuin H.323:n ja CSTA:n välillä. Sikäli, kun ohjausjärjestelmän osa toimii H.323:a käyttäen ja mahdolliset myöhemmät laajennukset SCTP:n pohjalta, olisi edessä näiden maailmojen yhteensovittaminen. SCTP-client on itse asiassa hyvin lähellä H.323-terminaalia ja vastaavasti SCTP-serveri H.323-gatekeeperiä, varsinkin SCTP-serveri suorittaa paljon gatekeeperille määriteltyjä tehtäviä. SCTP:n yhteydet ovat hyvin lähellä H.323:n informaatiovirtoja, joskaan eivät aivan identtisiä.

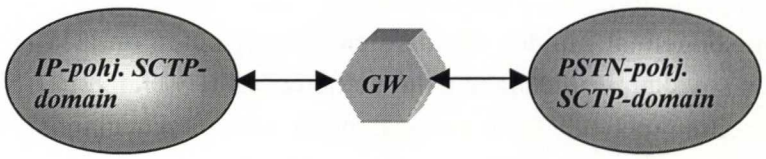
Jos SCTP-järjestelmä on liitettävä yhteen H.323-järjestelmän kanssa, se ei vaadi läheskään yhtä suurta työtä kuin CSTA-systeemiin liittäminen. Toistaiseksi on kuitenkin todettava, että kyseinen liittäminen tai siihen valmistautuminen on tarpeettoman työlästä, ja kuuluu jatkokehitykseen.

Täydennettynä SCTP voisi olla aivan riittävä yksinäänkin mediariippumattoman järjestelmän viestiprotokollaksi. Muutama seikka kuitenkin puhuu tätä vastaan heti ensi vaiheen toteutuksessa: SCTP:n kokeiluluontoisuus ja näinollen kypsymättömyys ja sitä tukevien järjestelmien puute vaikeuttavat sen lopullista testaamista. Toisaalta vaadittaisiin paljon työtä puuttuvien ominaisuuksien toteuttamiseksi, jotka kuitenkin valmistuessaan olisivat epästandardeja.

Koska tulevaisuus kuitenkin saattaa tuoda SCTP:lle saman volyymin kuin SMTP:lle kävi, käydään tässä lyhyesti lävitse, mitä SCTP:ltä lisäksi vielä vaadittaisiin, jotta saataisiin aikaiseksi esim. H.323:n tasoinen järjestelmä.

1) Usean serverin tuki / serverin valinta. SCTP olettaa vakioserverin, jonne tunnistaudutaan. H.323:ssa gatekeeper voidaan etsiä jostakin, jos vakio- tai useimmin käytetty gatekeeper ei ole palveluvalmiudessa [ITUT96]. Toki SCTP:n päälle voidaan tällaisia viestejä rakentaa, mutta ne eivät enää ole SCTP:n skoopissa.

2) Mediariippumattomuuden tuki. Vaikka SCTP ei suoranaisesti otakaan kantaa käytettyyn mediaan tai siirtotiehen, on se mietitty kuitenkin käytettäväksi ympäristössä, jossa varsinainen mediavirta kulkee PSTN-linjoja pitkin. SCTP sopii kyllä IP-ympäristöökin, mutta siinä tapauksessa se on puhtaasti IP-ympäristössä toimivaa, eli yhteydet PSTN-maailmaan ovat suljetut. Linjaoperaatiot sallivat ulkopuolisiin systeemeihin liittymisen, mutta vetävät rajan linjan alkuun. Esimerkiksi IP/PSTN-gatewayn toiminta on kokonaan SCTP:n ulkopuolella: em. gatewayn kytkeminen mukaan SCTP-järjestelmiin kävisi ainoastaan asentamalla se kahden erityyppisen SCTP-domainin väliin (kuva 3.11).



Kuva 3.11: IP/PSTN-gateway ja SCTP.

SCTP:hen kaivattaisiin siis sekä yleisempiä funktioita että lisää viestejä/funktioita mm. mediatyypin muunnoksiin.

3) Lisää viestejä gatekeeper-toiminnallisuuteen.

4) ACD-toiminnallisuuden viestit: johtuen SCTP:n suunnitteluperiaatteesta siinä ei ole minkäänlaisia käyttäjän tukiviestejä (ryhmittelyä ja soittoketjujen yms. rakentamista verten) eikä monimutkaisempia puhelunkäsittelyviestejä (esim. esto) parametrien vähyydestä puhumattakaan (esim. soittajan tietojen puuttuminen NewCall-viestissä)

3.6.2 NM-API – TAPI 3.0

TAPI 3.0:n merkitys CT-sovelluksille yleensä ei ole mitenkään mullistava – TAPI 2.1 riittää useimmille. Niille CT-sovelluksille taas, joilla on tarkoitus yhdistää eri medioita, tai mahdollisesti käsitellä niitä riippumattomalla tavalla, TAPI 3.0 tarjoaa yhden ensimmäisistä

rajapinnoista, joka todella ottaa huomioon kaikki usean median käsittelyyn liittyvät ongelmat. Siinä, missä NetMeeting ja sen API on tarkoitettu pieneen ja suhteelliseen yksinkertaiseen puhelun/mediavirran käsittelyyn, TAPI 3.0 tarjoaa täysimittaisen ohjelmointirajapinnan lähes kaikkeen kuviteltavissa olevaan mediavirralla operointiin.

Rajapintojen erot seuraavat niiden kautta käytettävien ohjelmien käyttötarkoituksia: NetMeeting on tarkoitettu lähinnä kahdenväliseen keskusteluun (audio, video & chat), yhdessä työskentelyyn ja tiedostojen siirtoon, lähinnä siis etätyöskentelyn mahdollistavaksi ohjelmaksi. NetMeeting-API:n kautta voi käynnistää konferenssin, irrottautua siitä, ja saada tiedot näistä tapahtumista. Muuten sillä voi enimmäkseen siirtää tiedostoja, ohjata chat-työskentelyä tai harrastaa sovellusten jakamista.

Tässä työssä käsiteltyyn aiheeseen nähden TAPI 3.0 olisi ehdottomasti parempi vaihtoehto. Valitettavasti koko TAPI 3.0 rajapintaa ei tätä kirjoitettaessa ole vielä julkistettu, puhumattakaan välttämättömistä MSP:stä ja TSP:stä. TAPI 3.0 kokonaisuudessaan on sen verran kunnianhimoinen ajatus (varsinkin suodingraafeineen), että sen ilmestyminen edes NT 2000:n myötä [MICR97c] voi olla liian positiivisesti ajateltu.

Tämän työn kannalta ainoa realistinen API-vaihtoehto on NetMeeting-API, johtuen juuri sen avoimuudesta ja laajasta käyttäjäkunnasta. NetMeeting-API ei edes ole huono vaihtoehto prototyyppiä varten, koska se on suhteellisen luotettava ja testattu, ja toisaalta se kuitenkin tukee riitävää perussettiä mediavirran ohjauskomennoista. TAPI 3.0:aa ja seuraavaa versiota odotellessa NetMeetingin API on täysin riittävä valinta päästä käsiksi H.323-yhteensopivuuteen.

4. Universal Service Queue

4.1 Määrittely

Universal Service Queue eli USQ tarkoittaa lyhyesti ohjausjärjestelmän kannalta saapuvien yleistettyjen puheluiden l. palvelupyyntöjen [D_{AWS}96a] käsittelylogiikkaa. Koska saapuvia eli inbound-pyyntöjä tulee yleisessä tapauksessa tiheämmin kuin niitä pystytään käsittelemään, on palvelupyynnöt varastoitava väliaikaisesti johonkin. Tämä välivarasto toteuttaa muutamin poikkeuksin jonologiikan, ns. FIFO-periaatteen. Puhtaassa PSTN-ympäristössä esimerkkinä ovat esimerkiksi vaihteen (PBX:n) sarjat. Kun välivarastossa voi olla useammantyyppisiä palvelupyyntöjä, esimerkiksi IP-puheluita, PSTN-puheluita ja sähköposteja [D_{AWS}96a], kutsutaan sitä USQ:ksi. USQ onkin kaikkein ahtaimmassa merkityksessään juuri tämä eri tahoilta saapuvien palvelupyyntöjen jonomuotoinen välivarasto. Laajemmassa merkityksessä USQ voi tarkoittaa kaikkia niitä serverikomponentteja, jotka mahdollistavat mediariippumattoman synkronisten palvelupyyntöjen käsittelyn. Tässä luvussa pelkkää palvelupyyntöjen jonoa kutsutaan USQ:ksi, ja serverikomponenttien joukkoa USQ-järjestelmäksi.

Koska eri medioita käsitellään hyvinkin erilaisin sovelluksin ja laitteistoin, ei niitä kannata puristaa saman putken läpi fyysisesti. Periaatteena on se, että USQ käsittelee palvelupyynnöt oman formaattinsa mukaisesti loogisella tasolla, ja käyttää standardirajapintoja palvelupyyntöjen mediaspesifisiin komponentteihin päin. Tällä tavalla ohjausjärjestelmältä on abstrahoitu media pois, eikä varsinaisiin mediavirtoja käsitteleviin komponentteihin tarvita muutoksia, korkeintaan joitakin lisäosia.

USQ:hun kuuluu muutakin kuin pelkkä mediariippumaton palvelupyyntöjen jono. Jonologiikka, l. jonoon valittavat palvelupyynnöt ja jonosta poistamisen kriteerit kuuluvat myös USQ:n piiriin. Jos USQ olisi pelkkä yksinkertainen FIFO-rakenne, ei jonologiikassa olisi juurikaan erikoisuuksia. USQ:n mediariippumattomuus tuo kuitenkin mukanaan erilaisia poikkeustilanteita, jotka johtavat tarkan jonokurin rikkomiseen paikka paikoin. Nämä määritellään tarkemmin toiminnallisuudesta kertovassa luvussa.

Tärkeä osa USQ:n toteutusta ovat rajapinnat. USQ:lla on oltava paitsi selkeät rajat, myös rajapinnat, jotta eri mediatyyppejä olisi mahdollista käsitellä. Rajapinnat nojaavat vahvasti PSTN-maailman ajattelutapaan ja käsitteistöön. Arkkitehtuurissa on esitelty USQ laajemmassa merkityksessään, koska eri serverikomponenttien identifioiminen on välttämätöntä itse toteutusta varten.

On huomattava, että USQ:n tyyppisiä järjestelmiä ei toistaiseksi ole juuri ollut olemassa [D_{AWS}96c, H_{ANN}96]. Tämän vuoksi edellä esiteltävä materiaalia ei juuri ole ollut mahdollista testata, ja määritystä on pidettävä enemmän mukautumiskykyisenä ehdotuksena, kuin lukkoonlyötynä kuvauksena. Joissakin kohdissa melko täsmällisesti ja yksityis-

kohtaisesti esitetyt asiat ovat sellaisia juuri jatkokehityksen helpottamiseksi. Pakollisiksi ja optionaalisiksi merkityt viestit saattavat muuttua, vaihtaa statustaan tai jopa poistua.

Koko järjestelmä olisi voitu konstruoida myös jonoteoreettiselta pohjalta, mutta useiden tähän liittyvien standardien, tekniikoiden ja teknologioiden kypsymättömyydestä johtuen itse USQ:kin on esitetty tässä toiminnallisesta lähtökohdasta paneutumatta tehokkuus-näkökohtiin. Näkökulma on sikäli tarvelähtöinen, että haetun järjestelmän *toimivuus* on tässä ensisijaisena kriteerinä, joskaan tässä esitetty määrittely ei sulje jonoteoreettisiakaan lisämäärityksiä pois.

4.1.1 Toiminnallisuus

USQ:n tulee pystyä vastaanottamaan, jonottamaan ja toimittamaan hallitsemansa tyyppiset palvelupyynnöt perille näiden mediasta riippumatta. Se, miten em. toiminnot hallitaan, on tarkemmin selvitetty toteutuksesta kertovassa luvussa. Tässä määritellään, miten USQ:n odotetaan toimivan loppukäyttäjän kannalta.

USQ voidaan konfiguroida hallitsemaan tietty joukko medioita. Näiden medioiden välillä USQ:n tulee pystyä yhtenäiseen käsittelyyn, ja mahdollisesti erikseen niin määriteltäessä, myös muunnoksiin eri medioiden välillä. Muunnokset tehdään tarvittaessa, eli käytettävissä olevan laitteiston ja sen arkkitehtuurin puitteissa: esimerkkitoteutus lähtee tapauksesta, jossa IP-puhelut ja PSTN-puhelut kulkevat erikseen, eikä niitä voida sekoittaa, mutta käsittely on yhteinen.

USQ-järjestelmässä voi olla useita riippumattomia loogisia jonoja, joissa palvelupyynnöt matkaavat. Jonoihin assosiotuja pyyntöjen käsittelijöitä on neljää tyyppiä. Käsittelijöitä kutsutaan:

- agenteiksi, jos niille voidaan *tarjota* puhelua, ts. voidaan lähettää asynkroninen pyyntö jollekin toiselle taholle palvelupyynnön vastaanottamisesta, ja saada kyseiseltä taholta myönteinen tai kielteinen vastaus, jonka perusteella palvelupyyntö voidaan poistaa jonosta. Tyypillisesti agentit toteuttavat lisäksi CSTA:n mukaisen määrittelyn agentista. USQ sisältää perustoiminnot agenttien ja ryhmien hallintaa varten.
- jononkäsittelijöiksi, jos ne vain esimerkiksi muuttavat pyynnön prioriteettia tai yhdistävät pyynnön väliaikaisesti jonnekin muualle, kuten IVR:ään tai IWR:ään.
- asiakkaaksi, jos ne pystyvät poistamaan pyynnön jonosta vaihdekomponentin ulkopuolelta, mutta kuitenkin tämän kautta. Ollakseen asiakas kyseisen käsittelijän on myös oltava osana kyseisen puhelun CSTA-kuvausta.

- ylivuotorutiiniksi, jos se voi poistaa puhelun jonosta olematta agentti tai asiakas.

Samaan jonoon assosioituja agentteja kutsutaan ryhmäksi tai palveluksi, ja interaktiivisia jonotuksenaikaisia viihdykkeitä tarjoavia järjestelmiä, kuten IVR:ää ja IWR:ää (Interactive Web Response, mm. $E_{FUS}97$) kutsutaan tässä yleisesti IUR:ksi (Interactive USQ Response).

Vastaanottaminen tapahtuu, kun jokin USQ-järjestelmään kuuluva vaihdekomponentti lähettää viestin **USQ_X_ALERTING** itse jonolle. Kyseinen viesti kertoo, että palvelupyyntö on saapunut vaihdekomponentille. USQ-järjestelmän jonoa edeltävät logiikkakomponentit päättelevät pyynnön tunnistetiedoista sen mahdollisen prioriteetin sekä sitä käsittelevän palvelun (jonon). Kun jono ottaa vastaan pyynnön, se vahvistaa viestin **USQ_X_ALERTING** vastaavalla vahvistusviestillä **USQ_X_ALERTING_CONF**, ja aloittaa prosessoinnin.

Jonotuksessa huomioidaan kolme asiaa: prioriteetti, skooppi, ja jononkäsittely. Prioriteetilla viitataan jonossa olevan pyynnön kiireellisyyteen: mitä korkeampi prioriteetti, sitä lähempänä jonon käsiteltävää päätä pyyntö on. Skoopilla tai näkyvyydellä tarkoitetaan sitä, mitkä jonon osat ovat näkyvissä kenellekin ryhmän jäsenille. Jononkäsittelyn mukaisesti jonottava pyyntö ohjataan fyysisesti esim. VRU-linjalle, IWR:een, tai annetaan pyynnön jonottaa täydellisessä "hiljaisuudessa". Jononkäsittely pitää myös sisällään virhetilanteiden käsittelyn.

USQ-järjestelmän oheislaitteiston mukaan skooppi on joko mediakohtainen tai mediariippumaton. Mediakohtaisesta skooppia tarvitaan, jos medianmuunnoslaitteistoa ei ole käytettävissä. Mediakohtaisessa skoopissa tyypin T puhelut näkyvät kaikille skoopin puitteissa oleville agenteille, jotka pystyvät käsittelemään tyypin T mediaa. Vastaavasti mediariippumattomassa mallissa kaikki puhelut/pyynnöt näkyvät (skoopin rajoissa edelleen) kaikille agenteille.

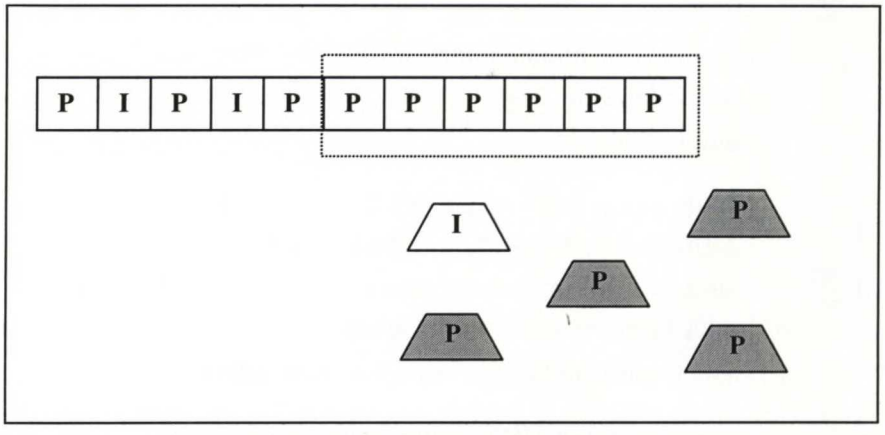
Skoopit jaetaan kahteen tyyppiin, `offer-to-all`, ja `offer-to-one`-puheluiksi. Skoopit ovat puhelukohtaisia, mutta tyypillisesti kaikki samaan jonoon saapuvat puhelut ovat samassa skoopissa. Tällöin jonoa/palvelua kutsutaan `offer-to-all` tai `offer-to-one` – palveluksi. `Offer-to-all`-puhelu tarkoittaa, että kaikki agentit "näkevät" sen puhelun, ts. kyseistä puhelua tarjotaan kaikkien agenttien käsiteltäväksi. Vastaavasti `Offer-to-one`-puhelua tarjotaan vain sille agentille, jonka vuoro on kyseinen puhelu käsitellä. `Offer-to-one` – tilanteita varten annetaan jokaiselle agentille RPN (Response Priority Number) jonka perusteella valitaan agentti, jolle puhelua tarjotaan. Mediakohtainen skooppi (ne agentit, keille puhelua/puheluita tarjotaan) määritellään seuraavasti:

- Muille kuin AGENT_FREE-tilassa oleville agenteille ei tarjota puheluita.

- Vapaalle (AGENT_STATE = AGENT_FREE) agentille tarjotaan niitä medioita sisältäviä puheluita, joita agentin työympäristö/muut määrittelyt mahdollistavat agentin vastaanottaa seuraavasti:
  - Puheluita tarjotaan jonon alusta sen loppuun saakka tai ensimmäiseen sellaiseen `offer-to-one`-puheluun, joka kyseisen agentin on määrä ottaa.
  - Muut `offer-to-one`-puhelut, kuin kyseessä olevalle agentille edellisessä kohdassa tarkoitetulla välillä osoitetut, eivät näy agentille. Vasta kun jokin jonossa edellä oleva puhelu "ylivuotaa", ts. agentti, jonka kuuluisi käsitellä kyseinen `offer-to-one` puhelu, ei siihen vastaa, voidaan puhelu siirtää uudelle agentille.
  - Agentilla oleva RPN määrää, monettako `offer-to-one` puhelua hänelle tarjotaan. Suurimman RPN:n omaavalle agentille tarjotaan ensimmäistä `offer-to-one`-puhelua, toiseksi suurimmalle RPN:lle toista, jne. Kun agentti vastaa johonkin offer-to-one-puheluun, hänen RPN:nsä nollataan ja muiden RPN:iä vastaavasti lisätään yhdellä.
- `Offer-to-one`-puhelu ei voi muuttua `offer-to-all` puheluksi, mutta päinvastoin voi tapahtua.
- `Offer-to-all`-puhelu muuttuu `offer-to-one`-puheluksi jos sen prioriteetti on riittävän suuri ollakseen jonon kärjessä ja vaatiakseen välitöntä huomiota. Tällä menettelyllä ehkäistään mahdollinen nälkiintyminen jonon kärjessä (ts. jos soittaja on tarpeeksi kärsivällinen, hänelle taataan pääsy käsittelyyn jossakin vaiheessa, mikäli ylivuotoparametrejä ei ole asetettu toisin)
- Jos ylläpitäjän määräämät ylivuotoehdot täyttyvät jossakin kohdassa jonotusta, puhelu siirretään pois jonosta ja näinollen myös agentin skoopista.
- Puhelu on agentin skoopissa silloin, kun se jonossa sijaitsee agentin kohdalle tarjotun `offer-to-one`-puhelun edessä. Mikäli agentin kohdalle ei ole yhtään `offer-to-one` puheluita tarjottuna, agentin skooppi on sama kuin koko jono, jolloin saapuva puhelu on välittömästi agentin skoopissa.

USQ:n agenteille tarjoama "ikkuna" voi olla ylläpitäjän määrittelemä, eikä tämä dokumentti ota kantaa laitteistossa tapahtuvaan jonottamiseen, jota logiikkakomponentit eivät näe. Jos ikkuna on pienempi kuin koko jono, on mahdollista, että mediakohtaisessa skoopissa ei näy sellaisia puheluja, jotka voitaisiin ottaa vastaan, mutteivät pääse

etenemään skoopin alueelle, koska edessäolevat puhelut tukkivat jonon. Tästä esimerkkinä kuvan 4.1 tapaus.



Kuva 4.1: Mediakohtaisen skoopin tukkeutuminen

Ylläolevassa kuvassa ikkunan koko on pienempi kuin varsinaisten jonottavien palvelupyyntöjen lukumäärä. Jos tyyppiä P käsittelevät agentit ovat kaikki varattuja, jono ei etene minnekään, vaikka tyyppiä I käsittelevä agentti olisi vapaana. Eteneminen tyypin I kannalta on pysähtynyt, koska tyyppiä I oleva puhelu ei näy kenellekään, eikä sitä siis voi myöskään tarjota yhdellekään agentille. Edellämainittua tilannetta ei pääse syntymään, mikäli ikkunan kooksi pakotetaan koko jonon koko.

Mediariippumaton skooppi toimii kuten mediakohtainenkin skooppi sillä erotuksella, että vaikka agenttien käyttämät mediat saattavatkin vaihdella, niillä ei ole merkitystä, joten edelläkuvatun kaltainen tilanne ei enää ole mahdollinen (myöskin pelkkää I-tyyppiä käsittelevä agentti olisi käsittelemässä yhtä P-tyypin puhelua, joskin yhdyskäytävän lävitse kulkeneena)

Agentin tila (AGE_FREE, AGE_DONOTDISTURB, AGE_OUT, AGE_BUSY) samoin kuin RPN voidaan asettaa **USQ_AGE_SETSTATE** – viestillä, tai kirjattaessa agenttia sisään.

Prioriteetti, eli jokaisen palvelupyynnön tärkeyttä kuvaava luku voidaan asettaa yhtäsuureksi vakioksi (=1) jokaiselle saapuvalle palvelupyynnölle. Tällöin kyseessä on perinteinen puhelujono. Jos prioriteetteja kuitenkin voidaan asettaa erisuuriksi, kyseessä on prioriteettijono, jossa suurimmilla prioriteeteilla saapuneet pyynnöt ovat lähimpänä käsiteltävää päätä. Jos ylläpitäjä on niin parametrisoinut, prioriteettia voi muuttaa, jos:

- Vaihdekomponentilta saapuvan palvelupyynnön tunnistetiedot (esim. A-numero) suhteessa tietokannassa olevaan tietoon niin edellyttävät
- Palvelupyynnön käsittelyaika on hupenemassa (esim. sähköpostille taattu vastausaika umpeutumassa [$D_{AWS}96a$, $H_{ANN}96$])

- Palvelupyyntö on jonottanut riittävän kauan jonossa pääsemättä yhdenkään agentin käsittelyyn.

Prioriteetti voi jossain tapauksissa muuttaa jopa jonon skooppia korottamalla `offer-to-all` puhelun `offer-to-one`-puheluksi, kuten skoopista kertovassa osuudessa on selvitetty.

Jononkäsittelyssä voidaan jonottava puhelu kytkeä fyysisesti muualle, vaikka se loogisesti onkin itse jonossa, tai luopua jonottamisesta kokonaan. Edelleen on erotettavissa mediakohtainen ja mediariippumaton jononkäsittely, jotka riippuvat käytettävissä olevasta laitteistosta/ohjelmista. Mediakohtainen jonokäsittely valitsee mediatyypeittäin erityyppiset viestit, mutta kommunikoi kuitenkin samojen USQ-rajapinnan viestien kautta muiden USQ-komponenttien kanssa. Esimerkiksi IP-puhelinta käyttävälle on luontevinta tarjota viihdykkeenä multimediaa, jos sitä on käytettävissä; PSTN-käyttäjälle taasen VRU:n äänitiedotteita. Lisäksi ylivuodon käsittelyyn voidaan tarjota enemmän tai vähemmän optioita mediasta riippuen.

Mediariippumattomassa järjestelmässä käytetyillä systeemeillä ei ole väliä, koska epäsopivat tapaukset voidaan tarvittaessa kääntää (Esim. IWR:stä suodattaa kuva pois ja ajaa koodaus IP-PSTN-yhdyskäytävän lävitse puhelinverkkoon) toisilleen sopiviksi.

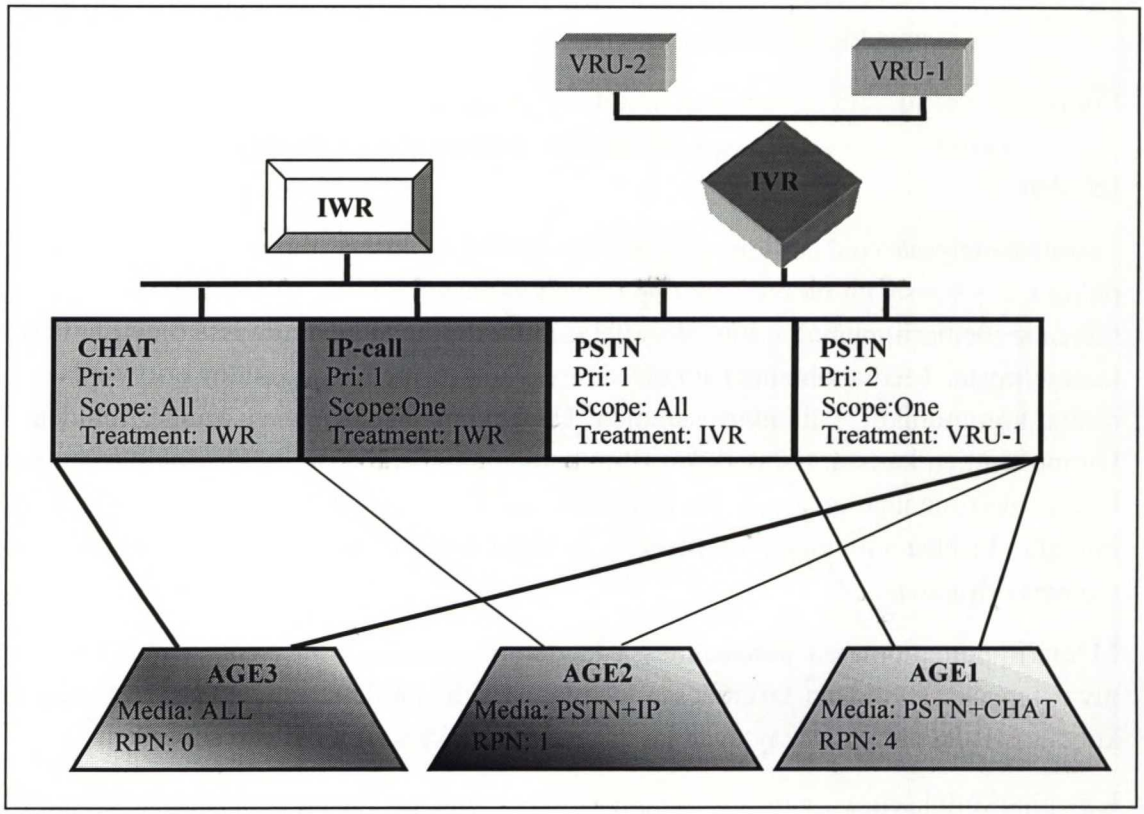
Käyttötarkoituksensa puolesta jononkäsittely voidaan jakaa ihmiskäyttäjälle informatiivisiin puhelunsiirtoihin tiedotteesta vastaavalle järjestelmälle sekä ylivuotostrategioihin. Ensinmainituista on PSTN-puolella olemassa VRU (Voice Response Unit) ja IVR (Interactive Voice Response), joista IVR on yleisempi, tyypillisesti VRU:ta hyväksikäyttävä järjestelmä. IP-puolelle on käytössä eräiden IP/PSTN-yhdyskäytäviä valmistavien firmojen (eFusion) lanseeraamat IWR (Interactive Web Response [$E_{FUS}97$]). Kaikkien näiden tarkoitus on pitää palvelupyyntöjen kysyjät "linjoilla" odottamassa.

Ylivuoto tapahtuu, kun järjestelmä päättelee (ylläpitäjän antamin parametrein) että ketään ei ole tavoitettavissa. Tämä päätelmä voidaan tehdä seuraavista syistä:

- Jono on täynnä
- Puhelu on jonottanut liian kauan
- Yhtään agenttia ei ole paikalla (AGENT_STATE = AGENT_OUT)

Ylivuodon tapauksessa palvelupyyntö poistetaan jonosta ja ohjataan jonnekin muualle. Tämän ylivuotopaikan voidaan spesifioida olevan sähköposti, puheposti, jokin päivystävä puhelin/pääte, tai jokin nauhoite.

Kuvassa 4.2 on vielä havainnollistettu jonottamiseen liittyvät käsitteet ja määritelty toiminnallisuus.

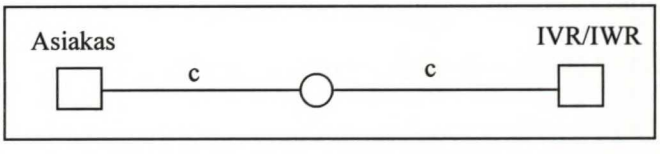


Kuva 4.2: USQ:n jononkäsittely, prioriteetti ja skooppi.

Agenttien skoopit on kuvattu niistä jonoon ulottuvin viivoin, ja jononkäsittelyn informatiiviset järjestelmät on piirretty itse jonon yläpuolelle zonen kautta käytettäväksi.

Perille toimittaminen tapahtuu jonottamisen jälkeen, joko ylivuotokäsittelyn tai agentin hyväksynnän tuloksena. Kun kohde on selvinnyt, USQ:n tulee selvittää pyynnön kohteen media ja mahdollisuudet käsitellä tulevaa mediaa sellaisenaan. Tämän jälkeen USQ-järjestelmän on joko asianmukaisen yhdyskäytävän kautta tai suoraan – mikäli lähteen ja kohteen mediat täsmäävät – siirrettävä puhelu loogisesta jonosta reaaliseen yhteyteen. Tässä voidaan käyttää virtuaalisia puheluita tarvittaessa hyväksi: tärkeintä on, että puhelu siirtyy loogisessa mielessä jonosta pois palvelupyynnön esittäjän ja agentin tai palvelupyynnön esittäjän ja ylivuototoiminnon väliseksi.

USQ:n täytyy myös pystyä käsittelemään erilaiset poikkeustilanteet, kuten palvelupyyntöön kohdistuvat asiakkaan suorittamat manuaaliset operaatiot. Normaali jonotustilanne on allaolevan kuvan 4.3 mukainen:



Kuva 4.3: asiakkaalle soitetaan jonoviestiä tämän jonottaessa

Koska asiakkaan puolen yhteys on vapaasti asiakaspään laitteen kontrolloitavissa, yhteyden (asiakkaan laitteesta puheluun) tila saattaa muuttua kesken jonotuksen connected-tilasta pidettyyn (h), tai kadota kokonaan (null = manuaalinen keskeytys, lopetus tai siirto [E_{CMA}94]). Jälkimmäinen tapaus on sikäli yksinkertainen, että USQ:n logiikkakomponenttien saadessa tästä tiedon vaihdekomponenteilta palvelupyyntö tarvitsee vain poistaa jonosta.

Asiakasyhteyden muuttuminen pidettyyn tilaan ymmärretään siten, että asiakas pitää yhteyttä yllä, muttei pysty kommunikoimaan. On epävarmaa, saako USQ ylipäätään tietää tämänkaltaisesta yhteyden tilanmuutoksesta. Tähän jätetään laitteistoriippuvaista toimintatilaa: jos toteutuksessa on mahdollista tietää asiakaspään yhteyden tila, voi USQ "jäädyttää" tämän palvelupyynnön prioriteetin siten, että tärkeämmät puhelut hyppivät pidetyn puhelun yli jonossa. Reagointi asiakaspään tilaan on siksi mediakohtaista, että kyseinen toiminnallisuus jätetään optionaaliseksi ja päätettäväksi mediakohtaisesti: jos asiakaspää pystyy näkemään jonon pituuden jostakin, voidaan olettaa, että asiakas pitää puhelua "pidettynä" sinne asti, kunnes jono hänen edellään kutistuu nollaan; toisaalta jos asiakkaan puhelulaite ei salli jononpituutta monitoroivaa toiminnallisuutta, ei voida myöskään olettaa asiakkaan ottavan puhelua pois pidosta agentin vastattua puheluun. Edellisessä tapauksessa "pidetty" puhelu on jonossa samanarvoinen muiden puheluiden kanssa, kun taas jälkimmäisessä sen ohitse voi vapaasti priorisoida muita puheluita.

Toiminnallisuuteen kuuluvat rajapinnat on esitetty kohdassa 4.1.2.

4.1.2 Rajat

4.1.2.1 USQ-järjestelmän komponentit

USQ-järjestelmän rajat jaetaan kahteen osaan: ydintoiminnallisuuden sisältävän itse USQ:n rajat, sekä USQ-järjestelmän rajat. Ensinmainittu käsittää myös rajapinnat USQ:sta muihin serverikomponentteihin päin, kun taas USQ-järjestelmä ei määrittele tarkemmin mediaspesifisten serverikomponenttien välisiä rajapintoja, eikä toisaalta ota kantaa ollenkaan serverikomponenttien ja ulkopuolisten järjestelmien välisiin rajapintoihin.

USQ-järjestelmään kuuluvat implementoinnin puolella esitetyt serverikomponentit, jotka voidaan jakaa neljään ryhmään:

- *Vaihdekomponentit*· Nämä ovat mediaspesifisiä, ja suorittavat USQ:ta yhtä abstraktiotasoa alemmat mediavirran ohjaustehtävät. Esimerkkinä PBX:t ja MCU:t.

- *Yhdyskäytäväkomponenttien* tehtävänä on suorittaa muunnokset mediavirran tyypistä X tyypiksi Y. Esimerkeiksi käyvät H.323-gatewayt, jotka suoritta-

vat PSTN-IP – muunnoksen; sekä puheentuotto- ja puheentunnistussovellukset, jotka voitaisiin periaatteessa valjastaa data – audio-muunnoksen yhdyskäytäviksi.

- *IUR-komponentit.* Näihin kuuluvat mediaspesifiset mediavirtaa tuottavat järjestelmät, joiden tarkoitus on jonotuksen aikana kommunikoida asiakkaan kanssa. IVR, IWR ja VRU ovat kaikki IUR-komponentteja.

- *Logiikkakomponentit*, joista jononkäsittelylogiikkaa prosessoivat osat muodostavat itse USQ:n. Muut logiikkakomponentit voivat läheisesti liittyä USQ-järjestelmään esim. puhelun jatko-ohjauksen tai resurssienhallinnan (agentit, laitteet, työasemat) puitteissa. Esimerkiksi gatekeeper (CustomerConnect-tilaserveri) on sellainen, joskin iso, logiikkakomponentti, joka pystyy hoitamaan USQ:n määritelmän vaatimat toiminnot.

Muut kuin em. neljässä ryhmässä olevat komponentit, esimerkiksi loppukäyttäjän client-komponentit, voidaan kyllä usein assosioida USQ-järjestelmään, ja ovatkin olennaisena osana mukana testicasen implementoinnissa, mutta eivät varsinaisesti kuulu itse USQ:n sisälle. USQ ei ota kantaa siihen, mikä sen lopullinen käyttötarkoitus on – ideaalitapauksessa USQ-toiminnallisuus, tai osia siitä voidaan istuttaa mihin tahansa muuhun järjestelmään middlewarena – joten USQ-järjestelmä kattaa vain em. serverikomponentit.

4.1.2.2 Vaihdekomponenttien välisistä rajapinnoista

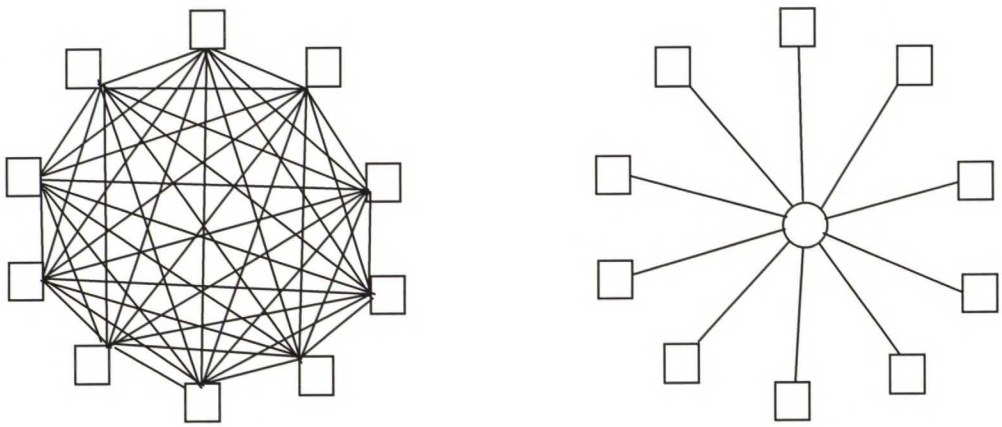
Vaihde- ja yhdyskäytäväkomponenttien välisestä kommunikoinnista annetaan vain hyvin määritellyt kehykset, jonka puitteissa kyseisten komponenttien tulee toimia. Seuraavassa pohdinnassa yhdyskäytäväkomponentit samaistetaan vaihdekomponentteihin.

Jokaisen vaihdekomponentin tulee mukautua tiettyyn korkean tason vaihdetoimintojen joukkoon. Tällaisia ovat mm. siirto (transfer), lopettaminen ja aloitus (hangup & initiate) ja vastaavat ilmoitukset kyseisistä toiminnoista, mikäli ne tapahtuvat muusta kuin vaihdekomponentin toimesta. Tämän implementointi onnistuu parhaiten rajapintoja periyttämällä. Koska vaihdekomponentit ovat hyvin sidoksissa perustana olevaan laitteistoon, on syytä käyttää jotakin vaihdeystävällistä mallia: tavallisella vaihteella (esim PBX:llä) on tyypillisesti vain jokin joukko muistipaikkoja, joiden välillä puhelua voidaan siirrellä. Näillä muistipaikoilla on myös jokin niihin assosioitu tila. Tästä voidaan johtaa yleinen joukko, joka kattaa kaikki mahdolliset puhelunohjaustoiminnot: olkoon ohjaustoimintojen joukko $M=\{D,S,T\}$, missä D on mahdollisesti jopa ääretön mutta kuitenkin numeroituva vaihteen käyttämien laitteiden joukko, S on äärellinen vaihteen sallimien tilojen joukko, ja T kokoelma funktioita $D^n x S^m \rightarrow D^p x S^q$, missä $n,p,m,q \in N \wedge n,p<L_d \wedge m,q<L_s$. L_s ja L_d ovat vaihdekohtaisia vakioita, jotka kertovat, kuinka monta mahdollista tilaa/laitetta kyseinen vaihde pystyy käsittelemään. Tätä joukkoa M käytetään seuraavasti: vaihdekomponentille annetaan jokin funktio $F \in T$, jossa tarvittavat laitteet ja tilat on järjestelmän puitteissa

yksikäsitteisesti identifioitu. Vaihdekomponentti riisuu niistä vaihde-ID:t pois ja välittää vaihdekohtaisen käskyn eteenpäin käyttämänsä laitteiston rajapinnan mukaisesti. Tässä järjestelmänlaajuisesta yksikäsitteisyydestä tehdään vaihteenlaajuinen, ja poistetaan yksi abstrahointikerros päältä.

Joukko M toteuttaa vaihdekomponentille tyypillisen riisutun CSTA-mallin kriteerit. Riisuttu malli ei käsitä kuin S-domainin funktiot, ja niissäkin puheluobjekti jätetään huomiotta. Toisaalta katsantokanta on 3rd party, ja laitteiden sekä yhteyksien tilat ovat kuten CSTA-mallissa.

Riisuttu CSTA-malli ei käsitä puheluobjekteja, koska vaihdekomponentti tuntee ainoastaan "laitteet", eli PBX/MCU:n sisäiset muistipaikat ja näiden väliset assosiaatiot. Tämä riisuttu malli ei silti ole yksinkertaisempi kun käsitellään vähänkään monimutkaisempia puhelunohjausoperaatioita. Tämä johtuu puheluobjektin yksi-yhteen kuvauksesta broadcast-tyyppiseen mediavirtaan: siinä missä 10 henkilön konferenssipuhelua joudutaan riisutulla mallilla kuvaamaan kymmenellä laitteella ja $(10^2+10)/2 = 55$ näiden välisellä assosiaatiolla (yhteydellä), riittää CSTA-mallissa yksi puheluobjekti, kymmenen yhteyttä ja kymmenen laitetta (kuva 4.4)



a) 10-välinen konferenssipuhelu riisutun CSTA-mallin mukaan

b) 10-välinen konferenssipuhelu CSTA-mallin mukaisesti

Kuva 4.4: Riisuttu CSTA-malli vs. CSTA-malli

Riisutussa CSTA-mallissa tarvitaan siis 44 oliota enemmän kuin varsinaisessa CSTA-mallissa. Tämä on kuitenkin S-domainin ominaisuus, koska vaihdekomponenttien ei kuiten kaan tarvitse tietää kokonaisuutta, ts. niiden ei tarvitse käsitellä kuin muutamien laitteiden joukkoja kerrallaan, siinä missä C-domainin komponenteille on tärkeää pystyä laskemaan funktioita suuresta kokoelmasta laite- ja yhteysobjekteja, jolloin on kustannustehokkaampaa käyttää CSTA-mallia kokonaan riisutun sijasta.

4.1.2.3 USQ:n rajapinta

Itse USQ käsittää ne edelläkerrotun toiminnallisuuden toteuttavat logiikkakomponentit, jotka toteuttavat ainakin seuraavassa taulukossa olevat pakollisiksi merkityt viestit (rajapinta on asynkroninen, mutta sitä voidaan käyttää tarvittaessa synkronisesti kuten RPC-pohjaista rajapintaa). On huomioitava X- ja GW-viestien yksinkertaisuus: nämä eivät kata läheskään kaikkia mahdollisia ohjaustoimintoja edellä esitetystä vaihderajapinnan ohjaustoimintojen joukosta, mutta kyse onkin vain jonottamisen vaatimista viesteistä.

Taulukko sisältää viestin nimen, lähettäjä- ja vastaanottajakomponentit, vaatiiko vahvistusviestin vastaanottajakomponentilta (E/K), ja onko optionaalinen USQ:n toteutuksessa. (Optionaaliset viestit eivät ole enää optionaalisia, jos perustana oleva laitteisto sallii viestin ilmaiseman toiminnallisuuden implementoinnin.) Viimeisessä sarakkeessa on selitys viestin tarkoituksesta.

Taulukko 4.1: USQ-rajapinta

Viesti	Lähettäjä	Vast.ottaja	ac.	Opt.	Selite
USQ_AGE_LOGIN	Client-komponentti	USQ:n logiikkakomp.	K	E	Agentti liittyy/liitetään USQ-systeemiin. Parametreina staattiset statusmerkinnät: ryhmäkuuluvuudet ja käytetyt mediat.
USQ_AGE_LOGOUT	Client-komponentti	USQ:n logiikkakomp.	K	E	Agentti poistuu/poistetaan USQ-systeemistä
USQ_AGE_SETSTATE	Client-komponentti tai agenteista huolehtiva logiikkakomp.	USQ:n logiikkakomp.	K	E	Agentin tila muuttuu (sis. CSTA-tilan, DnD:n, ryhmäkuuluvuudet ja käytetyt mediat) agentin tekemän toiminnon seurauksena
USQ_AGE_STATE-CHANGED	USQ:n logiikkakomp.	Broadcast kaikille siitä kiinnostuneille komponenteille.	K	E	Kuten ed. mutta tilamuutos johtuu serverikomponenteista.
USQ_GRP_ADD	Client-komponentti tai ryhmistä huolehtiva logiikkakomp.	USQ:n logiikkakomp.	K	E	Lisätään ryhmä USQ:hun. Parametreinä vakinaiset agentit ja käytetyt mediat, sekä skooppi-tagi.
USQ_GRP_DELETE	Client-komponentti tai	USQ:n	K	E	Poistetaan ryhmä/palvelu USQ:sta

	ryhmistä huolehtiva logiikkakomp.	logiikkakomp.			
USQ_GRP_EDIT	Client-komponentti tai ryhmistä huolehtiva logiikkakomp.	USQ:n logiikkakomp.	K	E	Muutetaan ryhmän kokoonpanoa tai tilaa. Tilaan kuuluvat skooppi (mikäli vakio) käytetyt mediat, sekä agentit tiloineen.
USQ_GW_CONNDELETE	USQ:n logiikkakomp./ vaihdekomp.	Yhdyskäytävätehtävistä huolehtivat serverikomponentit	K	K	USQ pyytää yhdyskäytäväkomponenttia lopettamaan yhteyden ja muuntamisen.
USQ_GW_CONNXFER	USQ:n logiikkakomp./ vaihdekomp.	Yhdyskäytävätehtävistä huolehtivat serverikomponentit	K	K	USQ pyytää yhdyskäytäväkomponenttia vaihtamaan yhteydessä olevaa osoitetta jatkaen kuitenkin muuntamista.
USQ_GW_CONNCREATE	USQ:n logiikkakomp./ vaihdekomp.	Yhdyskäytävätehtävistä huolehtivat serverikomponentit	K	K	USQ pyytää yhdyskäytäväkomponenttia ottamaan yhteyttä osoitteeseen, jonka tyyppiseen mediaan tämä yhdyskäytävä tarjoaa muunnospalveluja.
USQ_GW_CONNECT	USQ:n logiikkakomp./ vaihdekomp.	Yhdyskäytävätehtävistä huolehtivat serverikomponentit	K	K	USQ kytkeytyy yhdyskäytäväkomponenttiin (tunnistus ja mediaominaisuuksien vaihto)
USQ_GW_DISCONNECT	USQ:n logiikkakomp./ vaihdekomp.	Yhdyskäytävätehtävistä huolehtivat serverikomponentit	K	K	USQ irtautuu yhdyskäytäväkomponentista.
USQ_IUR_CONNECT	USQ:n logiikkakomp ./ vaihdekomp.	IUR-tehtävistä huolehtivat serverikomponentit	K	K	IWR/IVR/VRU – jonoviihdykettä tarjoavaan komponenttiin liittyminen. Parametrina komponentin linja, tms. tunniste, joka identifioi sen osan komponentista, jota tarvitaan.

USQ_IUR_DISCONNECT	USQ:n logiikkakomp ./ vaihdekomp.	IUR-tehtävistä huolehtivat serverikomponentit	K	K	IUR-komponentista poiskytkeytyminen.
USQ_IUR_START	USQ:n logiikkakomp ./ vaihdekomp.	IUR-tehtävistä huolehtivat serverikomponentit	K	K	Käsky IWR/IVR/VRU – jonoviihdykettä tarjoavalle komponentille aloittaa jonkin "kanavan" esittäminen – vaikkakin interaktiivisesti.
USQ_IUR_STOP	USQ:n logiikkakomp./ vaihdekomp.	IUR-tehtävistä huolehtivat serverikomponentit	K	K	Käsky IUR-komponentille lopettaa aloitetun kanavan esittäminen.
USQ_MEDIA_DEFINE	Client-komponentti	USQ:n logiikkakomp.	K	K	Ylläpitäjän client-komponentti määrittelee USQ:n käyttämät mediat.
USQ_MEDIA_OFFER-WINDOW	Client-komponentti	USQ:n logiikkakomp.	K	K	Ylläpitäjän client-komponentti määrittelee USQ:n jonossa agenteille tarjotun osan pituuden. (Näkyvän osan jonoa)
USQ_X_ALERTING	Vaihdekomp.	USQ:n logiikkakomp.	K	E	Vaihdekomponentilla on uusi palvelupyyntö tarjottavanaan USQ:lle, parametreina pyynnön tunnistetiedot.
USQ_X_ANSWERREQ	USQ:n logiikkakomp.	Vaihdekomponentti	K	K	USQ käskee vaihdekomponenttia vastaamaan puheluun (käytetään, kun puhelua ohjataan kutsunsiirroin, so. yhteys saattaa syntyyävasta, kun pyyntö on jo jonossa).
USQ_X_DEVICE_ADD	Vaihdekomponent ti tai laitteita ylläpitävä logiikkakomp.	USQ:n logiikkakomp.	K	E	Lisätään dynaamisesti uusi laite (terminaaliominaisuudet omaava) USQ-järjestelmään, parametreina tunnistetiedot ja mediaominaisuudet)
USQ_X_DEVICE_DEL	Vaihdekomponent ti tai laitteita ylläpitävä logiikkakomp.	USQ:n logiikkakomp.	K	E	Poistetaan dynaamisesti laite järjestelmästä.
USQ_X_DEVICE_EDIT	Vaihdekomponent ti tai laitteita	USQ:n	K	E	Muutetaan laitteen parametrejä, kuten mediaominaisuuksia. Laite kytketään

	ylläpitävä logiikkakomp.	logiikkakomp.			tällöin tilapäisesti irti järjestelmästä.
USQ_X_DIVERT	USQ:n logiikkakomp.	Vaihdekomponentti	K	E	USQ käskee vaihdekomponenttia siirtämään kutsun toiseen laitteeseen.
USQ_X_HANGUPREQ	USQ:n logiikkakomp.	Vaihdekomponentti	K	E	USQ käskee vaihdekomponenttia lopettamaan yhteyden (jonottavaan) asiakkaaseen.
USQ_X_MAN_ANS-WERED	Vaihdekomp.	USQ:n logiikkakomp.	E	E	Vaihdekomponentti tiedottaa, että agentti on vastannut pyyntöön käyttäen jotakin muuta käyttöliittymää, kuin järjestelmän tarjoamaa.
USQ_X_MAN_CONN-CLOSED	Vaihdekomp.	USQ:n logiikkakomp.	E	K	Kuten USQ_X_MAN_ HUNGUP, muttapätee yleisempiinkin häiriötilanteisiin.
USQ_X_MAN_GOT-FROMHOLD	Vaihdekomp.	USQ:n logiikkakomp.	E	K	Kuten USQ_X_MAN_GOTFROMHOLD, mutta muutos on h->c.
USQ_X_MAN_HUNGUP	Vaihdekomp.	USQ:n logiikkakomp.	E	E	Vaihdekomponentti tiedottaa, että joko asiakas on sulkenut yhteyden kesken jonotuksen, tai IUR:ssä on tapahtunut jotakin, joka on sulkenut yhteyden.
USQ_X_MAN_PUTON-HOLD	Vaihdekomp.	USQ:n logiikkakomp.	E	K	Vaihdekomponentti tiedottaa USQ:lle, että yhteys on pistetty "pidettyyn" (h) tilaan. Jonottavan pyynnön ollessa kyseessä ainoastaan asiakkaan pää voi tehdä tämän muutoksen, eikä se välttämättä tule perille USQ:lle.
USQ_X_SETOVER-FLOWTREATMENT	Client-komponentti	USQ:n logiikkakomp.	K	E	Järjestelmän ylläpitäjä asettaa ylivuoto-parametrejä, jotka kulkevat viestin mukana.
USQ_X_SETREQUEST-TREATMENT	USQ:n logiikkakomp.	Vaihdekomponentti	K	K	USQ käskee voihdekomponenttia asettamaan pyynnölle

					jonotusparametrit.
USQ_X_TRANSFER-REQUEST	USQ:n logiikkakomp.	Vaihdekomponentti	K	E	USQ käskee vaihdekomponenttia siirtämään puhelun haluttuun paikkaan.

Ylläoleva taulukko esittää viestipohjaista rajapintaa. Tämä on sikäli korkeamman abstraktiotason rajapinta, että viestien ympärille voidaan rakentaa objektirajapinta, kuten on laita esim. NetMeeting-API:ssa [M_{ICR}97b]. Luvussa 4.1.1 esilletulleista asioista ylivuodon ja jononalkion hallintaa ei ole otettu mukaan rajapintaan, koska ne ovat USQ:n logiikkakomponenttien sisäisiä tehtäviä.

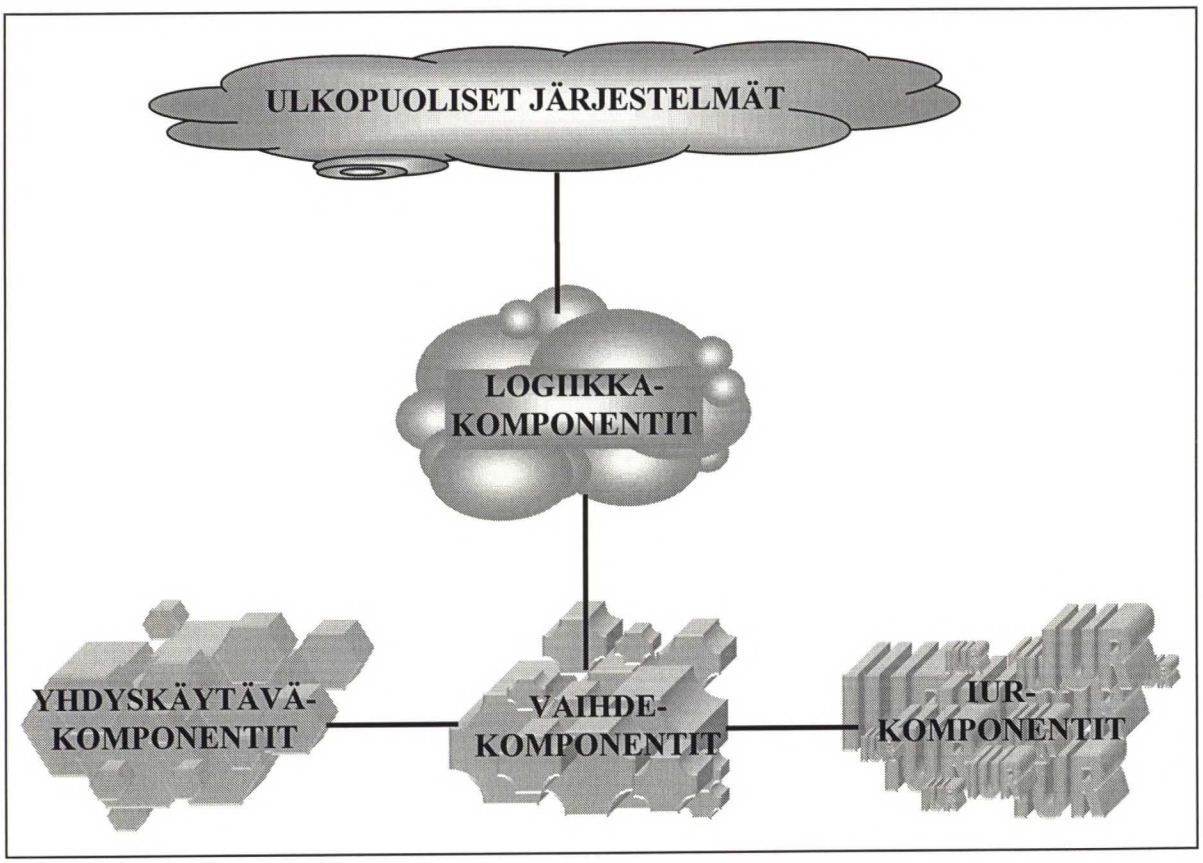
4.2 Toteutus

Edellä rajattu järjestelmä toimintoineen määrittelee ahtaasti katsoen varsinaisen USQ-järjestelmän. Toteutuksella ei USQ-järjestelmän kannalta ole suurta merkitystä, mutta yhteensopivaksi järjestelmän tekee oikea implementointi. Seuraavaksi esitetään ensin mahdollisimman yleispätevä arkkitehtuuri, jotta olisi helpompi ymmärtää testijärjestelmän komponenttien tarkoitus. Tämän jälkeen, luvussa 4.2.2, siirrytään asteittain yksityiskohtaisempaan ja samalla vähemmän yleispätevään arkkitehtuuriin – IP-PSTN – arkkitehtuuriskenaarioihin.

4.2.1 Yleinen arkkitehtuuri

USQ-järjestelmän arkkitehtuuri nojaa vahvasti PSTN-maailman käsitteistöön. Tämä taas ei implikoi CSTA:n käyttöä, koska se ei juurikaan ota kantaa arkkitehtuuriin, vaan pikemminkin reaalimaailmasta lainattuja PSTN-komponentti- ja laitenimiä yleistettynä. Syy PSTN-käsitteistön omaksumiseen mediariippumattomissa kommunikaatiojärjestelmissä on itse viestintätilanteen analogisuus puhelinkeskustelun kanssa – ainakin ensimmäisinä testattavissa point-to-point – yhteyksissä [D_{AWS}96a]. Niinpä USQ-järjestelmässä erotetaan, kuten rajauksessa on jo mainittu, *vaihdekomponentit, yhdyskäytäväkomponentit, IUR-komponentit* ja *logiikkakomponentit*.

Logiikkakomponentit ovat mediariippumattomia osia, ja kommunikoivat erikseen määriteltyjen viestien avulla muiden serverikomponenttien kanssa. Lisäksi USQ-järjestelmän ulkopuolista komponenteista kommunikoidaan mm. clienttien ja tietokannan kanssa, kuitenkin niin, että jonotuksesta huolehtivat komponentit abstrahoivat mediakohtaisten komponenttien, kuten vaihdekomponenttien olemassaolon kokonaan clienteilta.

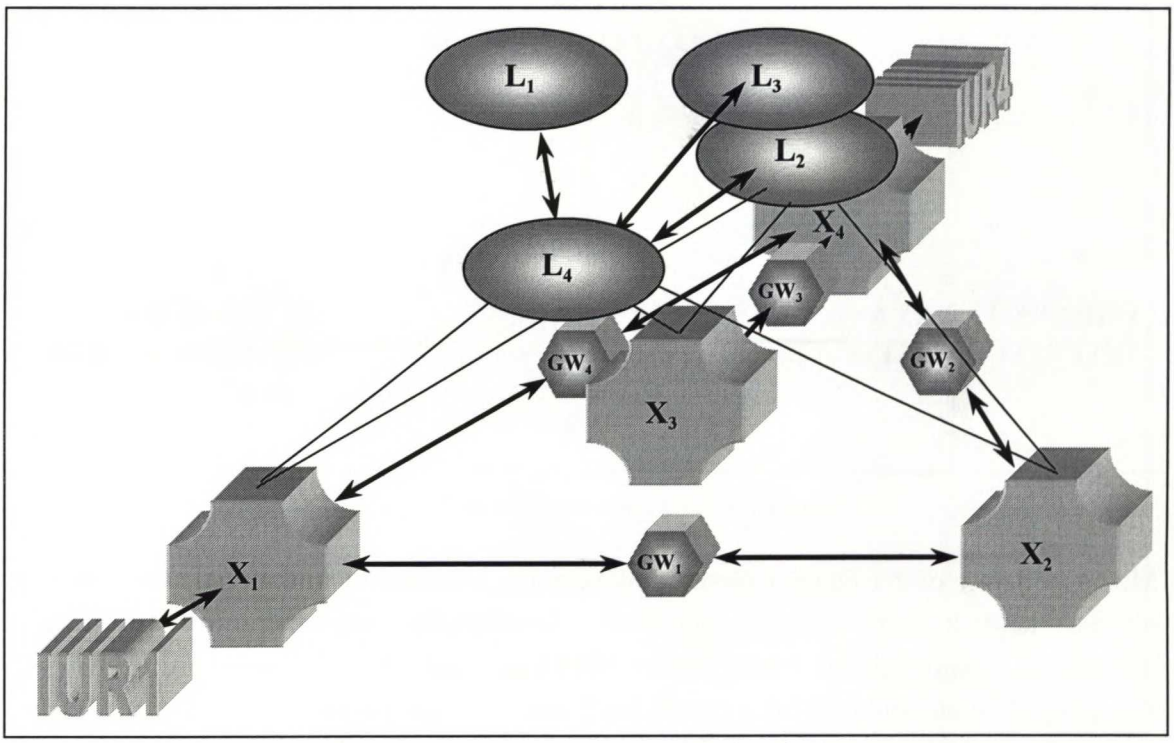


Kuva 4.5: USQ-arkkitehtuurin pilvimalli

Mediaspefisiset osat esitetään tässä edelläkerrotun luokittelun mukaisesti. Useimmissa jo olevissa käytännön toteutuksissa ja myös tulevaisuuden implementoinneissa useat nyt erilliset laitteistot tulevat yhdistymään luultavasti jopa samalle piirikortille [$H_{ANN}96$, $P_{ULV}98$]. Tämän mukana tulevien yleistettyjen API:den myötä tietyt mediat yhdistyvät keskenään riippumattomiksi virroiksi, mutta yhdistymisen jälkeenkin on muita medioita, jotka ovat edelleen erillään "valtavirrasta". Toiminnallisuutensa puolesta nyt esitettävä arkkitehtuurikuvaus ei ole tarkoitettu muutettavaksi, korkeintaan joitakin noodeja voidaan yhdistää samaksi kokonaisuudeksi.

Yleinen arkkitehtuuri voidaan nähdä pilvinä erityyppisiä komponentteja: vaihdekomponenttien pilvi keskustelee yhdyskäytävä- ja IUR-komponenttien pilvien kanssa, ja logiikkakomponenttien pilvi vaihdekomponenttien pilven sekä ulkopuolisten järjestelmien pilven kanssa. Tässä komponentin mielletään tarvittaessa sisältävän vastaavan laitteiston: PSTN-vaihdekomponentit PBX:n, PSTN-IP-yhdyskäytävät jonkin parin DSP-kortteja ja prosessoreja, PSTN-IUR-komponentit (IVR:t) vastaavat VRU-kortit, jne. Yleinen arkkitehtuuri on esitetty kuvassa 4.5.

Esimerkkinä siitä, mitä komponentteja logiikkapilvessä tarvitsee vähintään olla, sekä arkkitehtuurin toiminnallisuudesta esitetään kuvan 4.6 mukainen neljän median järjestelmä, jossa mukana on käytettävissä kaksi IUR-systeemiä, ja yhdyskäytäviä joidenkin medioiden välillä. Tässä kaksi vaihdekomponenttipilven vaihdetta on yhdistetty kahteen IUR-pilven komponenttiin, ja yhdyskäytäväpilven komponentteja on käytössä vaihteiden X_1-X_4, X_1-X_2, X_2-X_4 ja X_3-X_4 välillä. Mukaanotetut mediat ovat kysytyimpiä mediatyyppejä esim. yhdenmukaisestetuissa CallCenter-palveluissa [D_{AWS}96c, H_{ANN}96], joskaan medioiden välisiä yhdyskäytäviä ei ole vielä olemassa middleware-tarkoitukseen kuin IP/PSTN-tyyppisiä [D_{AWS}96b, P_{ULV}98]. Eri yhdyskäytävätyyppejä on lisätty lähinnä esimerkin vuoksi – havainnollistamaan USQ:n arkkitehtuurin toimintaperiaatteita.



Kuva 4.6: esimerkki eräästä USQ-järjestelmästä

Kuvaan ei ole piirretty komponenttien taustalla olevia laitteistoja, eikä myöskään asiakkaan tai agentin käytössä olevia laitteita tai ohjelmistokomponentteja. Agentin puolen komponentit laitteineen ovat tyypillisesti yhteydessä kunkin mediatyypin vaihteeseen, ja samoin - tapauskohtaisesti - myös asiakaspään sovelluskomponentit. Symbolit vastaavuuksineen on esitetty taulukossa 4.2:

Taulukko 4.2: USQ:n esimerkkiarkkitehtuurin symbolit

Sym-boli	Komp.luokka	Toiminta
L_1	Logiikka	Prioriteetin hallintakomponentti, antaa pyynnöstä prioriteetin jonotusalkioille, sekä valvoo näiden muutoksia. Sovellus-pohjainen.
L_2	Logiikka	Ylivuotojen hallintakomponentti, yhteydet vaihde-komponentteihin ylivuotojen toteuttamiseksi. Sov.pohjainen
L_3	Logiikka	Agenttien ja ryhmien hallintakomponentti, mm. sisäänkirjautuminen, ja vastaavat tietorakenteet. Sovelluspohjainen.
L_4	Logiikka	Jononhallinta. Keskeisin logiikkakomponentti: ottaa vastaan palvelupyynnöt vaihdekomponentilta, ja jonottaa sen sekä jakelee agenteille.Sovelluspohjainen.
X_1	Vaihde	H.323-määrityksien mukainen MCU. Sisältää MP:n, joka vuorostaan tarvitsee joukon DSP-kortteja. [$P_{ULV}98$]
X_2	Vaihde	Private Branch Exchange (PBX):n päällä oleva komponentti-sovellus. Varsin paljon laitteistoa alla.
X_3	Vaihde	MAPI-serveri, email-käsittelyjärjestelmä, tyypillisesti toimii erillisellä tietokoneella.
X_4	Vaihde	IRC-serveri, sovelluspohjainen.
IUR_1	IUR	IWR, MCU:hun kiinnitetty sovelluspohjainen web-IUR-järjestelmä.
IUR_2	IUR	IVR, muutaman DSP-kortin päällä toimiva Voice-IUR-systeemi
GW_1	Yhdyskäytävä	MCU:n ja IRC-srv:in välinen yhdyskäytävä. Näitä ei vielä ole kaupallisessa middleware-käytössä [$P_{ULV}98$]. Vaatii puheentunnistukseen kykenevän neuroverkon tai sellaista simuloivan ohjelman, ja äänisyntetisaattorin (äänikortin) sovellusohjelmineen. Kyettävä lisäksi suodattamaan mahd. video pois ja tarvittaessa lisäämään IRC-käyttäjän kuva IP-

		videon paikalle.
GW_2	Yhdyskäytävä	PBX:n ja IRC-srv:in välinen yhdyskäytävä. Näitä ei vielä ole kaupallisessa middleware-käytössä [$P_{ULV}98$]. Vaatii puheentunnistukseen kykenevän neuroverkon tai sellaista simuloivan ohjelman, ja äänisyntetisaattorin (äänikortin) sovellusohjelmineen.
GW_3	Yhdyskäytävä	PBX:n ja MAPI-srv:in välinen yhdyskäytävä. Ei vielä kaupallisessa middleware-käytössä [$P_{ULV}98$], vaatimukset samat kuin GW_2:lla, mutta neuroverkko opetettava vain rajalliseen määrään äänen tunnistusta → tulokset parempia. Lukee viestin kuin nauhurilta, kirjoittaa puheen tekstiksi.
GW_4	Yhdyskäytävä	MCU:n ja PBX:n välinen yhdyskäytävä, kaupallistettuina, mm. VocalTec, eFusion. Tarvitsee joukon DSP-kortteja, ja tyypillisesti oman prosessorin ja muistia käyttääkseen yhden PC:n resurssit kokonaan.

Jos edelläkuvatussa USQ-järjestelmässä tulee palvelupyyntö esimerkiksi X_4:n (IRC-serveri) kautta, tapahtuu seuraavaa:

- X_4 lähettää L_4:lle, eli jononhallintakomponentille **USQ_X_ALERTING** – viestin, jonka tämä vahvistaa vastaanottaneensa **USQ_X_ALERTING_CONF**-viestillä.
- L_3 - prioriteetinhallintakomponentti - päättelee tunnistetiedoista, kuten pyynnön lähettäjän IP-osoitteesta ja mahdollisesta tekstistä pyynnössä, päästetäänkö pyyntöä ylipäätään jonoon, ja mikä sen prioriteettinumeroksi annetaan.
- Jos pyyntöä ei päästetä jonoon, sille annetaan ensin tiedote IUR:n kautta, ja sen jälkeen suljetaan yhteys. Yhteyden sulkee L_4 lähettämällä **USQ_X-HANGUP** – viestin X_4:lle, joka katkaistuaan yhteyden lähettää **USQ_HANG-UP_CONF** -viestin takaisin L_4:lle.
- Jos pyyntö pääsee jonoon saakka, L_4 tutkii seuraavaksi sille määritellyn jonokäsittelyn: yhdistetäänkö se IUR:ään ja jos, niin mihin ja millä viestillä. (Kutsunsiirto jonotusnumeroon on välttämätöntä ainoastaan PBX:llä, joissa on rajoitettu määrä muistipaikkoja saapuville jonotusnumeroille).
- Jos valittu IUR on esimerkiksi IUR_1, on ensin kytkeydyttävä yhdyskäytävän GW_1 kautta X_1:een, ja annettava vasta sille komennot IUR:n soittamisesta.

Laitteiston integraatioasteesta ja ohjelmoitavuudesta riippuen L_4:kin voi joutua lähettämään nämä viestit (passiivinen yhdyskäytävä ja vaihde), mutta tässä oletetaan täysin kypsä systeemi.

- L_4 lähettää **USQ_X_DIVERT**-viestin X_4:lle, jossa osoitteena IUR_1
- X_4 lähettää **USQ_GW_CONNECT**in GW_1:lle, mikäli yhteys GW_1:een ei ole ennestään päällä, ja saatuaan tältä **USQ_GW_CONNECT _CONF**in, lähettää edelleen **USQ_GW_CONNCREATE**n IUR_1:n osoitteella
- GW_1 päättelee osoitteen olevan X_1:n alueella ja lähettää sille saapuvan puhelun signaalin IUR_1:n osoitteella.
- X_1 tunnistaa IUR_1:n osoitteen ja lähettää IUR_1:lle **USQ_IUR-CONNECT**in, mikäli yhteys IUR_1:een ei ole ennestään päällä.
- Kun X_1 on saanut vastauksen IUR_1:ltä **(USQ_IUR_CONNECT _CONF)**, X_1 siirtää mediavirran IUR_1:n linjalle ja käskee IUR:ää **USQ_IUR-PLAY**:llä aloittamaan mediavirran tuottamisen.
- Jos IUR vastaa **USQ_IUR_PLAY_CONF**illa X_1:lle mediavirran käynnistyneen odotetusti, X_1 vastaa edelleen GW_1:lle yhdistämällä mediavirran yhdyskäytävän X_1:lle tarjoamalle kanavalle.
- GW_1, havaittuaan, että yhteys on päällä, suorittaa muunnoksen ja palauttaa X_4:lle **USQ_GW_CONNCREATE_CONF**in, joka vuorostaan vastaa L_4:lle **USQ_X_DIVERT_CONF**illa, ja jonotiedote tulostuu GW:n kautta IRC-clientin päätteelle.

- L_4 keskustelee L_1:n (prioriteetti) ja L_3:n (agentinhallinta) kanssa siitä, kenelle kaikille pyyntöä pitäisi tarjota. Jos yhtään sopivaa agenttia ei löydy, puhelu ylivuodetaan luovuttamalla kontrolli L_2:lle.
- Jos agentti on hyväksynyt tarjotun puhelun, L_4 selvittää L_3:n kanssa mitä osoitetta agentti sillä hetkellä käyttää. Kun osoite on saatu selville, lähetetään X_4:lle jälleen uusi viesti, **USQ_X_TRANSFERREQUEST** agentin osoitteella.
- Jos X_4 päättelee agentin osoitteen olevan omalla alueellaan, siirto on yksinkertainen sen omien muistipaikkojen päivitys. Muussa tapauksessa käynnistetään IUR-yhteyden kaltainen medianmuuntorutiini sille vaihteelle, jossa X_4 uskoo osoitteen olevan, tai jollei se tunnista osoitetta antaa sen ”kykenevimmälle” vaihteelle, joka on siihen kytkeytyneenä.

- IUR-yhteyden purku tapahtuu seuraavasti (mikäli agenttiyhteys on eri mediaa: jos agenttiyhteyden ja IUR-yhteyden media sopivat toisiinsa, ei yhteyttä pureta X_1:tä pidemmälle, jolloin X_1 hoitaa yhteyden luomisen agenttiinsa):
  - X_4 lähettää **USQ_GW_CONNDELETE**n GW_1:lle IUR_1:n tunnisteella
  - GW_1 päättelee tunnisteen olevan X_1:n alueella ja lähettää sille yhteydenpurkusignaalin IUR_1:n tunnisteella.
  - X_1 tunnistaa IUR_1:n osoitteen ja lähettää IUR_1:lle **USQ_IUR_STOP**in
  - Kun X_1 on saanut vastauksen IUR_1:ltä (**USQ_IUR_STOP_CONF**), ja kun mediavirran tuotanto lakannut, kutsutaan **USQ_IUR_DISCONNECT** IUR_1:lle, mikäli muita yhteyksiä ei ole päällä.
  - Kun X_1 on lopettanut mediavirran siirtämisen GW_1:lle, tämä välittää **USQ_GW_CONNDELETE_CONF**in GW_1:lle, jolloin muiden yhteyksien puuttuessa tämä kutsuu **USQ_GW_DISCONNECT**ia GW_1:lle.

Edellä kuvattu jonotustapahtuma ohittaa koko joukon yksityiskohtia, esim. yhdyskäytävän ja vaihteen välisen kommunikoinnin ja "kyvykkäimmän" vaihteen etsimisen, jos vaihde ei löydä kohdemediaa tulkkaamaan pystyviä yhdyskäytäviä suoraan olemassaolevilta yhteyksiltä. Näihin ei puututtu, koska useamman kuin kahden median systeemit ovat toistaiseksi yhdyskäytävien puutteen takia suhteellisen kartoittamattomia [H_{ANN}96, P_{ULV}98]. Yleinen arkkitehtuuri pätee kuitenkin esimerkkitapauksineen varsin kattavasti. Tarkemmin em. yksityiskohtia tutkitaan IP-PSTN-arkkitehtuurissa, koska tässä tapauksessa olemassaolevat järjestelmät tarjoavat kokeellisen pohjan teorialle.

4.2.2 IP-PSTN arkkitehtuuri

Edellä on kuvattu täysin mediariippumattoman USQ-järjestelmän arkkitehtuuri ja yleisin viestintätapahtuma. Toistaiseksi varsinaiseen toteutukseen ei ole tämän työn puitteissa sopivia työkaluja kuin IP:n ja PSTN:n välille. IP-PSTN-arkkitehtuuriin on tarjolla jo useita erilaisia yhdyskäytäväkomponentteja, tosin hyvin erilaisia modularisuudeltaan ja ohjelmoitavuudeltaan [H_{ANN}96, P_{ULV}98]. Serverikomponenttien ja laitteiston arkkitehtuuri riippuu hyvin paljon käytetystä yhdyskäytävästä. Tarkastelussa ei puututa aluksi logiikkakomponenttien sijoitteluun, koska ne noudattavat edellä esitettyä yleistä ratkaisua. Sen sijaan keskitytään laitteistoläheiseen tarkasteluun, jonka pohjalta konstruoidaan täysimittainen IP-PSTN – arkkitehtuuri.

Yhdyskäytävä on tyypillisesti rakenteeltaan serveritason PC (väh. 200 MHz:n Pentium ja yli 64 MB RAM), johon on liitetty yksi tai useampia DSP-kortteja. Liityntäpinnat IP-verkkoon päin ovat yleensä ATM-pohjaisia suurinopeuksisia kaapeleita, ja vaihteeseen/

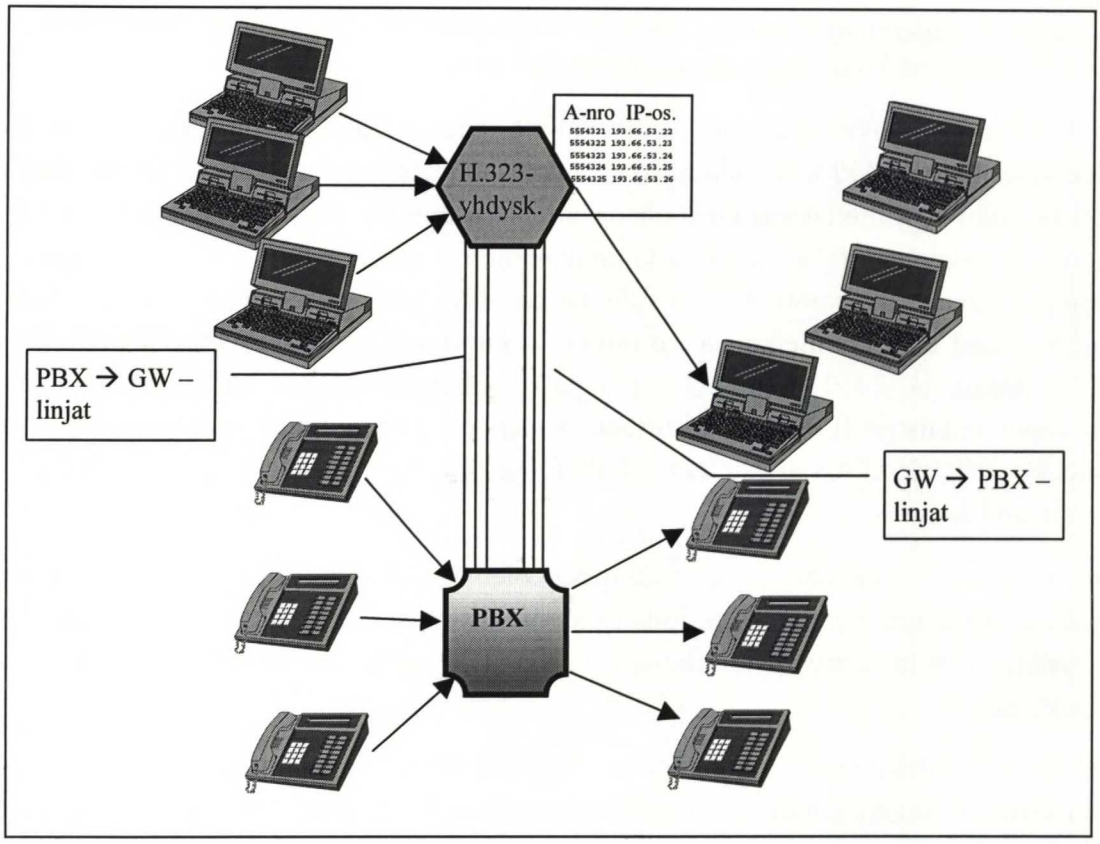
televerkkoon päin ISDN-liittymä (E1 tai PRI) [$D_{IAL}97$, $E_{FUS}97$]. Signaalin purku ja koodaus voidaan hoitaa joko DSP-korteilla olevassa muistissa, kuten **MICOM**in *V/IP*-järjestelmässä, tai CPUn käytössä olevalla muistilla, kuten esim **eFusion**in *eStream* ja **VocalTec**in *Telephony Gateway* tekevät. Nämä vaativat joitakin megatavuja yhtä mediavirran muunnosta kohti, joten skaalautuvuus ei ole kovin hyvä.

Yhdyskäytävän täysimittainen hyväksikäyttö vaatisi jonkin objektitason rajapinnan tukemista, kuten TAPI3, tai vähintään CSTA-tason puhelunohjauskäskyjä hyväksikäyttävää API:ta. Näin ei valitettavasti vielä ole asianlaita: useimmat yhdyskäytävät tukevat vain sen ympärille rakennetun järjestelmän komponenttien välisiä rajapintoja. Osa ei pysty vielä minkäänlaiseen dynaamiseen ohjattavuuteen [$P_{ULV}98$]. Äänenlaadussa ja koodausalgoritmeissä on vielä melkoista variointia, joskin trendi on H.323-yhteensopivia järjestelmiä kohtaan, ja TAPI3:n myötä myös palvelunlaatuun tullaan kiinnittämään enemmän huomiota erilaisten IP-verkon resursseja varaavien menetelmien muodossa. Seuraavassa esitetään erilaisia USQ-järjestelmän IP/PSTN-arkkitehtuuriskenaarioita eritasoisten yhdyskäytävien kanssa.

Ensimmäisenä skenaariossa on käytössä olemassaolevan yhdyskäytävän (**eFusionin** *eStream*) kaltainen komponentti, jolla ei ole julkistettua API:ta, mutta jota voidaan käyttää kuitenkin jonkinasteiseen toiminnallisuuteen [$E_{FUS}97$]. Arkkitehtuuri on kuvan 4.7 mukainen.

Kuvan 4.7 esittämässä arkkitehtuurissa yhdyskäytäväkomponentti ja vaihdekomponentti sijaitsevat fyysisesti samoissa tiloissa. Käytännössä tämä johtuu PBX:n ja yhdyskäytävän välisestä rajapinnasta, joka on kaapelipohjainen. Tämä aiheuttaa melkoisia rajoituksia USQ:n hajautettavuuteen, mitä tulee yhdyskäytävän ja PBX:n väliseen kommunikointiin. IP-vaihdekomponentti – MCU – voi kyllä sijaita missä tahansa, kunhan se on IP-verkon päässä, mutta PBX:n ja yhdyskäytävän välimatkan rajaavat käytetyt laitteistot ja rajapinnat varsin lyhyeksi.

Kuvassa 4.7 yhdyskäytävä toimii kyllä molempiin suuntiin, mutta kukin suunta on valmiiksi konfiguroitu kullekin linjalle. IP-PSTN-muunnoksessa yhdyskäytävän jokainen linja on kiinnitetty johonkin vaihteen alanumeroon. Kaikki yhdyskäytävään tulevat IP-puhelut ohjataan staattisesti johonkin em. vaihdenumeroista. Kun PBX on saanut puhelun, se käsittelee sitä kuin mitä tahansa muuta puhelua. IP-puheluita voidaan tässä tapauksessa tehdä suoraan yhdyskäytävään, jolloin se voidaan määritellä tekemään PIN- / IP-osoitetunnistus. (Tämä ei kylläkään ole välttämätöntä, sillä kiinnitetyillä PSTN-numeroilla kukaan ei pääse tekemään yhdyskäytävällä muita puheluita kuin minne on tarkoitettu). Yhdyskäytävän ne linjat, jotka tekevät yhdyskäytävästä vaihteen ”jatkeen”, avustavat vaihdetta siten, että vaihde voi oikealla konfiguraatiolla tehdä kutsunsiirron yhdyskäytävälle, joka sen jälkeen katsoo reititystaulustaan, mihin IP-osoitteeseen kyseisessä vaihdenumerossa oleva puhelu pitäisi konvertoida.



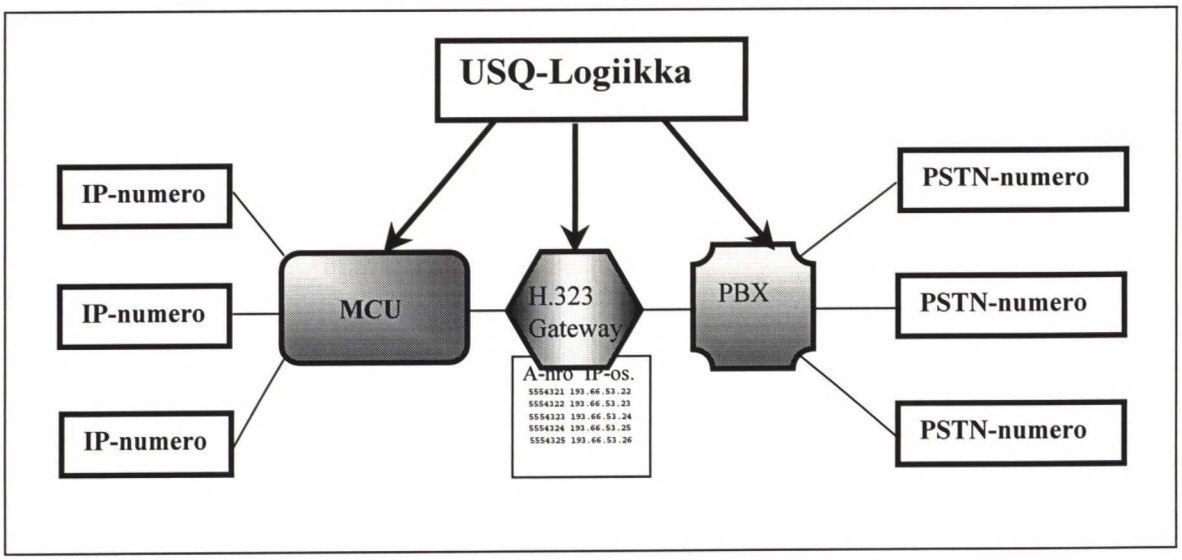
Kuva 4.7: Staattisesti konfiguroitava USQ: serveriarkkitehtuuri

Kyseiseen reititystauluun voi ylläpitäjä upottaa joitakin ohjaustietoja hallinnointirajapinnan kautta. PBX-yhdyskäytävä-linjat on piilotettu ulkomaailmalta, ja niihin pääsee vain esimerkiksi soittamalla johonkin PBX:n alanumeroon, joka voi huomattuaan, että kyseinen numero on esimerkiksi varattu, siirtää puhelun yhdyskäytävän numeroon, joka edelleen tekee siirtotiemuunnoksen ja yhdistää puhelun kyseistä puhelinta lähimpänä olevalle päätteelle [EFUS97].

Sikäli, kun vailla API:ta oleva GW pystytään yhdistämään PBX:ään ja MCUhun kiinteillä linjoilla, ja nämä linjat voidaan konfiguroida suorittamaan aina jonkinsuuntainen median-muunnos, voidaan päästä jo USQ:n vaatimaan toiminnallisuuteen. Arkkitehtuuri oletetaan kuvan 4.8 mukaiseksi.

Tässä toimintojen periaatteena on kääriä yhdyskäytävä tavittaessa MCU:n ja PBX:n väliin: jos tarvitaan IP-PSTN-muunnosta, puhelu siirretään MCU:lta PBX:lle yhdyskäytävän kautta. On huomattava, että yhdyskäytävän käyttämät sisäiset numerot ovat vakioita. Ainoastaan näiden välisiä kytkentöjä voidaan tehdä hallinnointirajapinnan kautta reititystauluun [E_{FUS}97]. Varsinainen puhelunohjaus tapahtuu vaihdekomponenteissa, joita vuo-

rostaan komentavat USQ:n logiikkakomponentit. Yhdyskäytävien komentamisesta huolehtivat vaihdekomponentit jonkin käskyrajapinnan kautta.

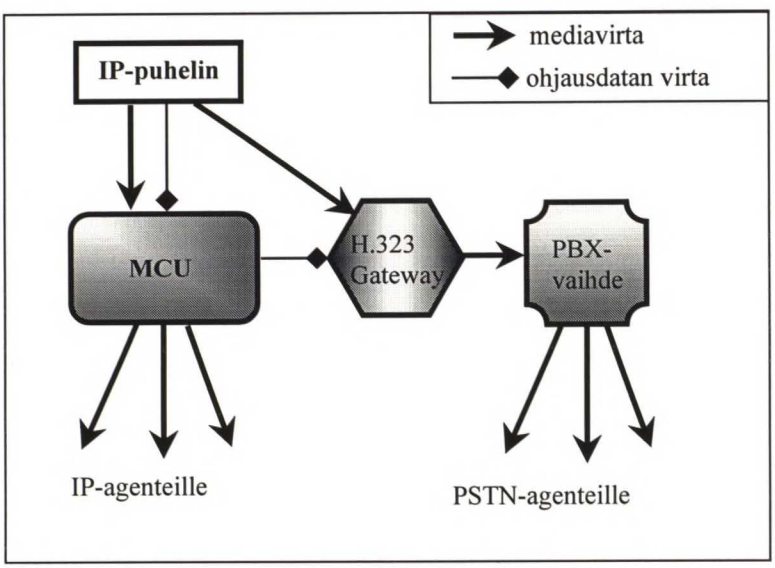


Kuva 4.8: Ilman APIa olevalla yhdyskäytävällä operoiminen

Ylläoleva järjestelmä pystyy, jopa ilman MP:tä olevan MCU:n kanssa (nykyiset USQ-tarkoitukseen sopivat MCU:t eivät sisällä MP:tä) toteuttamaan USQ:n vaatiman minimitoiminnallisuuden pystyen myös sekoittamaan eri puhelutyyppejä.

***Tapaus 1a*: saapuva palvelupyyntö IP-mediakanavaa pitkin**

Jos IP-muotoinen palvelupyyntö saapuu systeemiin, ensimmäinen sen näkevä vaihdekomponentti on MCU. Jos logiikkakomponentit määräävät kohteeksi PSTN-puhelimen, MCU vastaa asiakkaan IP-puhelimelle yhdyskäytävän IP-osoitteella, johon on liitetty PBX:n yhdyskäytävälle tälle linjalle osoittama numero.



Kuva 4.9: IP-mediana saapuvan palvelupyynnön käsittely.

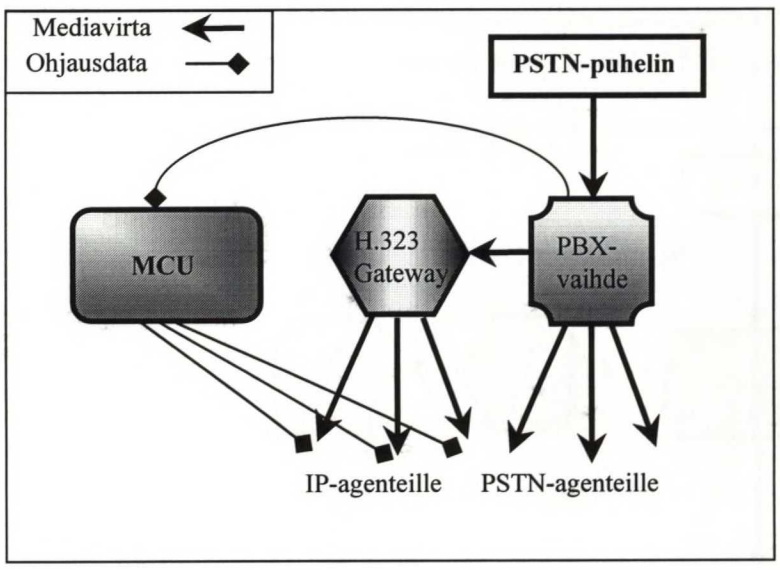
Asiakkaan IP-puhelin soittaa tämän jälkeen tähän numeroon (esim. 193.66.53.233, 09580-45012), jonka jälkeen PBX näkee sen tavallisena saapuvana puheluna, ja käsittelee sitä tälle yhdyskäytävän linja-numerolle asetettujen parametrien mukaisesti. Pelkät IP-to-IP-puhelut kulkevat suoraan MCU:n lävitse. Kuva 4.9 selventää tilannetta.

On huomattava, että koska MCU:sta puuttuu MP, mediavirta ei kulje koskaan MCU:n kautta, vaan joko suoraan IP-puhelimelta toiselle (asiakas-agentti) tai IP-puhelimelta yhdyskäytävälle. Ainoastaan ohjausdata liikkuu puhelimelta MCU:lle.

***Tapaus 1b*: saapuva palvelupyyntö PSTN-mediakanavaa pitkin**

PSTN-puhelut, mikäli logiikkakomponentti päättää niitä ohjattavan IP-agenteille, on laitettava kulkemaan myös yhdyskäytävän kautta. Mikäli mediamuunnosta ei tarvitse tehdä, PBX käsittelee PSTN-puhelut, kuin koko IP-mediaa ei olisi olemassakaan. Jos puhelu kuitenkin tarvitsee muuntaa IP-streamiksi, ovat vaadittavat operaatiot ilman API:ta olevan yhdyskäytävän kanssa suhteellisen työläitä, ja vaativat ehkä liikaakin erikoistoimintoja laitteilta, joita ei tähän käyttöön ole tarkoitettu.

Koska yhdyskäytävä, vaikkei API:ta tarjoakaan, sisältää kuitenkin jonkinlaisen käyttöliittymän, sen automaattinen käyttäminen PSTN-maailmasta osoittautuu vaikeaksi. Tarvitaan yksi ylimääräinen digitaalinen laite (käytännössä digitaalinen puhelin, digitaaliset ääntä prosessoivat laitteet tuntuvat olevan kiven takana, ja vaihde taas on analoginen), koska analogiset laitteet eivät pysty käsittelemään vaadittuja operaatioita.



Kuva 4.10: PSTN-mediana saapuvan palvelupyynnön käsittely.

Kun PSTN-puhelu saapuu PBX:ään, ja sen kohteeksi on määrätty IP-osoite, PBX:ää kontrolloivien komponenttien on seuraavaksi määrättävä tämä ylimääräinen digitaalinen laite ottamaan yhteys yhdyskäytävän IP-puolen kiinteään numeroon ja tuottamaan ns. key-string (näppäilysarjaa vastaava DTMF-sarja) kohteen IP-osoitteesta yhdyskäytävän linjalle. Jälleen on muistettava, että ei tuoteta MCU:n key-stringiä, koska tässä ei ole MP:tä, eikä näin ollen voi prosessoida mediavirtaa. Toiminta karkealla tasolla on esitetty kuvassa 4.10.

Kun PBX-komponentti on siirtänyt puhelun yhdyskäytävälle oikean IP-osoitteen kera, sen täytyy vielä informoida MCU:ta näistä muutoksista, jonka jälkeen MCU voi ottaa vastuun prosessista komentamalla puheluun osallistuvien agenttien ja asiakkaan IP-puhelimia. PBX:stä vastuussa olevan komponentin hallitseman siirron yksityiskohdat PSTN-puhelimen ja yhdyskäytävän välillä ovat varsin monimutkaiset:

1) PSTN-laitteella kommunikoiva asiakas soittaa ryhmän loogiseen numeroon. Tämä jonotetaan, ja kun päätetään kyseisen puhelun siirtämisestä IP-agentille, se siirretään yhdyskäytäväoperaatioille omistettuun jononumeroon (GWQ).

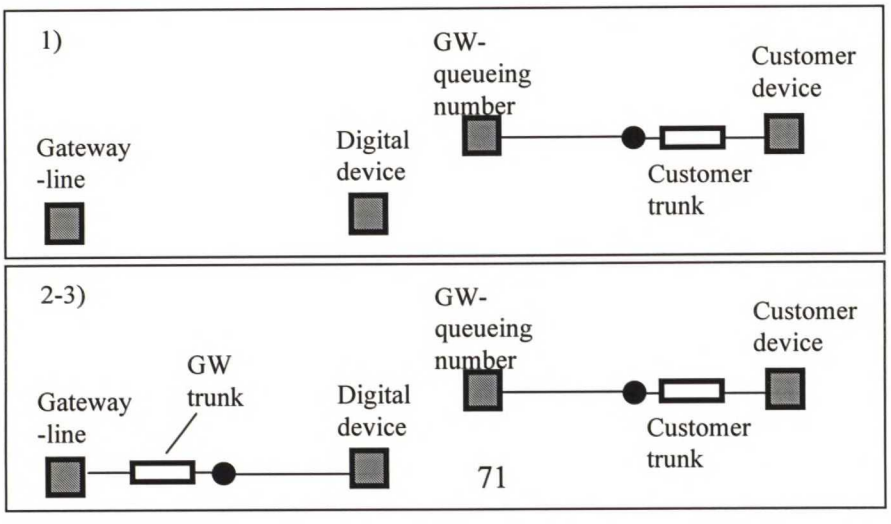
2) Ylimääräinen digitaalinen laite määrätään ottamaan yhteys yhdyskäytävään, ja synnytämään CSTA-yhteys sinne.

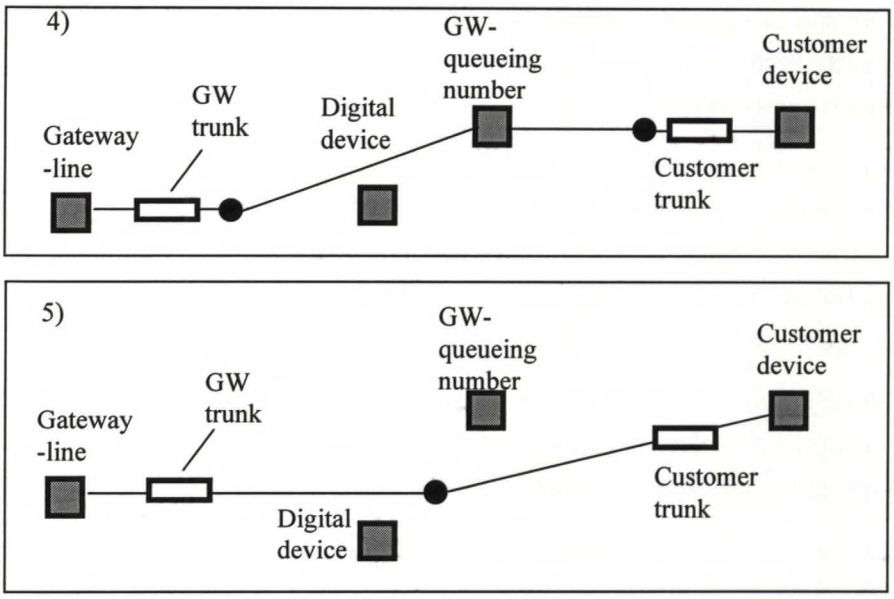
3) Digitaalinen laite komennetaan syöttämään kohdeagentin IP-osoitteen key-string yhdyskäytävään.

4) Digitaalinen laite pudotetaan pois siirtämällä CSTA-yhteys yhdyskäytävä-digitaalinen laite – väliltä yhdyskäytävän ja PBX:n GWQ:n välille. Tämä tehdään CSTA:n mukaisen välikysely-operaation avulla. (Syy, miksi analogista laitetta ei voinut käyttää on juuri välikysely-operaation digitaalisuus)

5) GWQ pudotetaan pois siirtämällä CSTA-yhteys yhdyskäytävän ”rungosta” (trunk, PBX:ssä ja yhdyskäytävän PBX-osassa oleva muistipaikka, johon voidaan yhdistää puhelu) PBX:ssä olevaan asiakkaan ”runkoon”. Nyt asiakkaan ja yhdyskäytävän välillä on vain yksi CSTA-puheluobjekti.

   Toiminnot on esitetty kuvasarjassa 4.11





Kuvasarja 4.11: PBX-yhdyskäytävä-siirto

***Tapaus 2*: Puhelun jatkokäsittely**

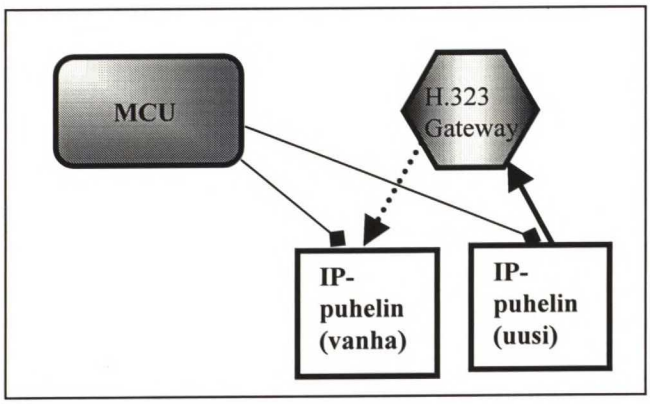
Vaikka puhelun jatkokäsittely ei periaatteessa kuulu enää itse USQ:n tehtäviin, sen alla olevan laitteistoarkkitehtuurin tulee silti pystyä mukautumaan myös näihin toimintoihin, koska kyseiset laitteet ovat liian kalliita hankittavaksi pelkästään USQ:ta varten. Tyypillisesti yksinkertaiset toiminnot, kuten hangup, onnistuvat helposti. Ongelmakohdaksi osoittautuu nimenomaan puhelun jatkoreitittäminen, jos se jo kulkee useampien laitekomponenttien kautta. Erityisesti puhelunsiirto osoittautuu mielenkiintoiseksi.

Puhelunsiirrot yhdyskäytävän sisältävässä järjestelmässä voidaan jakaa kahteen luokkaan: ne siirrot, joissa puhelu alunperin kulkee yhdykäytävän lävitse, ja ne jotka eivät kulje. Viimeksimainittu tapaus muodostuu identtiseksi saapuvan puhelun käsittelyn kanssa (jonotuksen jälkeen): joko siirto tapahtuu saman median sisällä, tai se vaatii yhden muunnoskomponentin reitilleen. Kummatkin tapaukset on käsitelty edellä. Toinen tapaus pitää sisällään joko yhdyskäytävästä luopumisen – ja paluun yhden median piirissä tapahtuvaan kommunikointiin – tai puhelun siirtämisen yhdyskäytävän toisella puolella.

Yhdyskäytävästä luopuminen ei sisällä kuin yksinkertaisen siirron, joka sulkee yhteyden yhdyskäytävään. Sekä MCU että PBX pystyvät tekemään nämä siirrot vakiotoiminnallisuudellaan tekemättä yhdyskäytävälle mitään.

Yhdyskäytävän sisältävä siirto on tällä kertaa vaikeampi, jos siirrossa toisena osapuolena olevat puhelimet ovat IP-puhelimia. (Siirto siis tapahtuu: PSTN-IP$_1$ → PSTN-IP$_2$), kuin jos

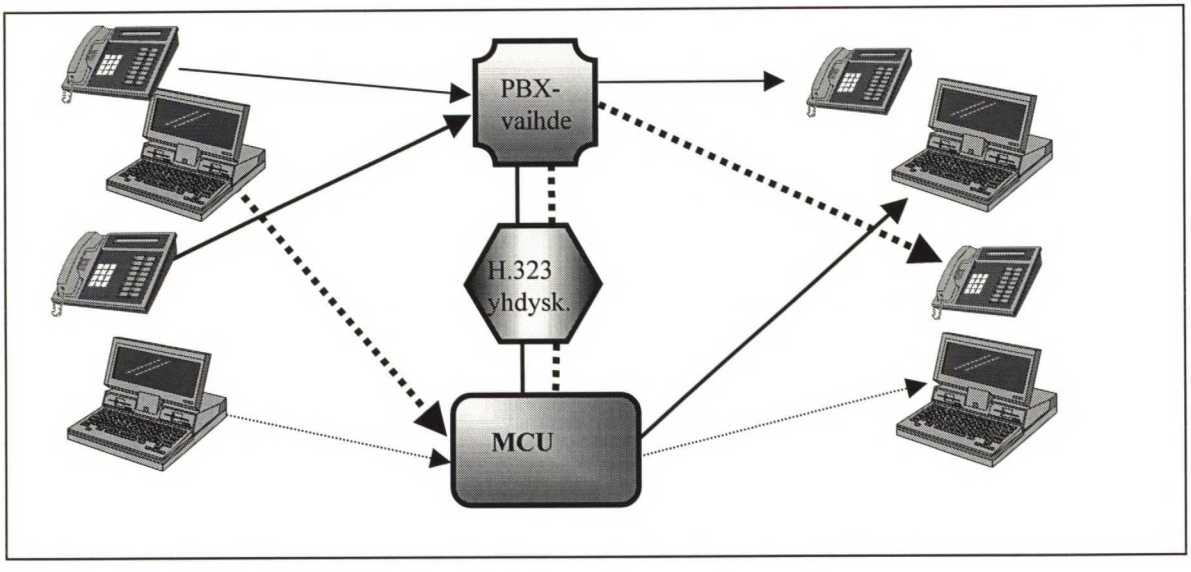
ne olisivat PSTN-puhelimia (Tarkasteltava siirto tällöin: IP-PSTN$_1$ → IP-PSTN$_2$). Ensimmäisessä tapauksessa, l. IP-siirrossa, MCU komentaa IP$_2$:n osallistumaan yhdyskäytävän isännöimään konferenssiin ja IP$_1$:n jättämään kyseisen konferenssin (Kuva 4.12).



Kuva 4.12: IP-puhelunsiirto (PSTN-IP$_1$ → PSTN-IP$_2$)

PSTN-siirto ei taas oikeastaan ole yhdyskäytävästä riippuva operaatio: koska puhelu sen aloituksen jälkeen kulkee yhdyskäytävästä PBX:ään (joka toisin kuin MCU pystyy käsittelemään mediavirtaa), pystytään mediavirralle tekemään mitä vain PBX pystyy tekemään, mm. siirtämään se toiseen osoitteeseen (numeroon).

Jos yhdyskäytävällä on vaihdekomponenttien käytettävissä oleva API, ei ole enää merkitystä sillä, ovatko linjat staattisesti kytkettyjä vaiko eivät. Tällöin kyseessä on edellisessä luvussa esitetyn esimerkkiarkkitehtuurin osajoukko, joka IP:n ja PSTN:n osalta on täysin siirtotie-/mediariippumaton. Laitteistoarkkitehtuuri näyttää nyt kuvan 4.13 mukaiselta.

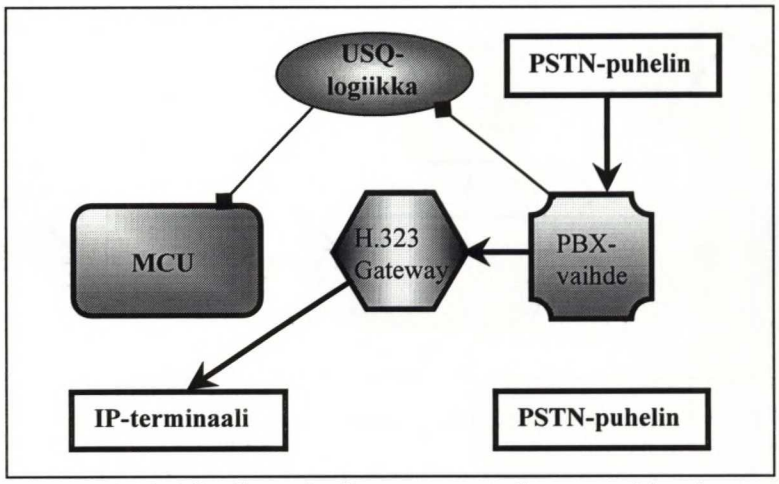


Kuva 4.13: Yhdyskäytävä vaihdekomponenttina muiden joukossa.

Kuvassa 4.13 mallinnetussa hybridiarkkitehtuurissa PBX ja yhdyskäytävä ovat edelleen oletuksena saman kaapeloinnin päissä (käytännössä samoissa tiloissa) [EFUS97]. Jos MCU sisältää keskitetyn signaaliprosessoinnin MP:n muodossa – mediavirta kulkee MCU:n kautta – yllä esitetty arkkitehtuuri on myös tekninen kuvaus. Jos MCU ei sisällä signaaliprosessointia, vaan se pystyy ainoastaan kontrolloimaan mediavirtaa terminaalien (PC:t) kautta, ylläoleva kuvaus on vain loogisen tason näkemys asiasta. Teknisellä tasolla mediavirta kääntyykin tällöin jo ennen MCU:ta yhdyskäytävästä IP-terminaaleihin, ja vastaavasti saapuu IP-terminaaleista jo yhdyskäytävälle MCU:n sijasta. MCU yhdentyy yhdyskäytävään siinä mielessä, että mediavirta kulkee yhdyskäytävän kautta, mutta MCU kuitenkin kontrolloi tuota mediavirtaa.

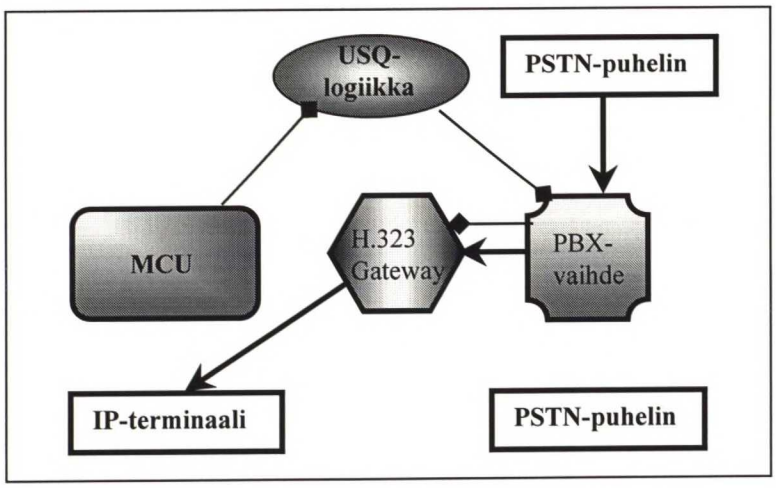
Kuvassa 4.13 PSTN-puhelimilla soitetaan PBX:n numeroihin, ja IP-terminaaleilla MCU:n osoitteisiin. Jos USQ:n logiikkakomponentit niin päättelevät, puhelu ohjataan yhdyskäytävän kautta toisen puhelutyypin vaihteelle, joka edelleen hoitaa sen eteenpäin oman tyyp-pisiinsä puhelinlaitteisiin. Myöhempi puhelunohjaus tapahtuu sillä periaatteella, että se vaihde, jonka alueella ohjauskomento tapahtui, informoi sitä vaihdetta, jonka alueella on staattisena pysyvä osapuoli, tekemään tarvittavat operaatiot. Tässä on kuitenkin otettava huomioon edellä kerrottu puhelunsiirtoskenaario yhdyskäytävän kautta: jos puhelu kulkee jo yhden kerran yhdyskäytävän lävitse, ei sitä tarvitse eikä saa ohjata takaisin sen lävitse: tämä on paitsi tehotonta, myös yhdyskäytävän resursseja kuluttavaa. On myös syytä abstrahoida yhdyskäytävä kokonaan pois logiikkakomponenttien näkyvistä. Tämä auttaa modulaarisuuden lisäksi eräisiin puhelunohjausoperaatioihin, esim. yhdyskäytävän läpi tapahtuvaan siirtoon (MCU:ssa ei oleteta olevan MP:tä):

1) Jos haluttu siirto on PSTN-IP → PSTN-PSTN, pyyntö tulee joltakin PSTN-puhelimelta PBX:lle, joka tiedottaa siitä eteenpäin logiikalle. Logiikka etsii nykyisestä päätepisteestä vastuussa olevan vaihdekomponentin, ja tiedottaa sille siirrosta. Kyseinen vaihdekomponentti on MCU, joka pistää siirron ylös omiin tietorakenteisiinsa (Kuva 4.14)



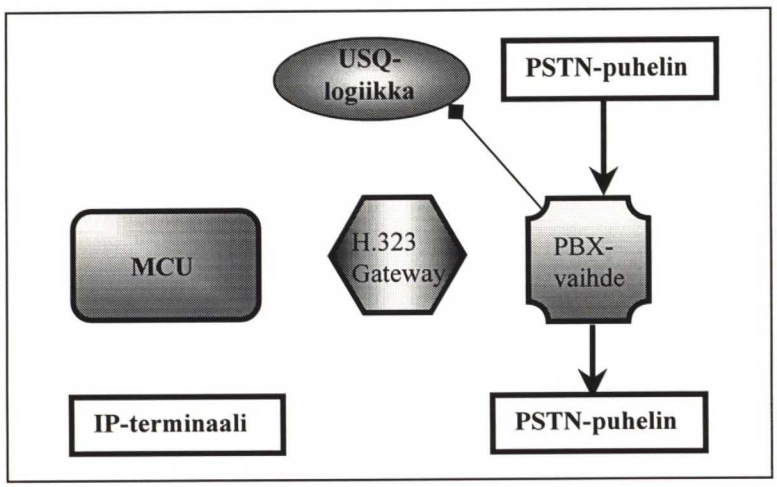
Kuva 4.14: Alkutilanne ja siirtopyynnöstä aiheutuvat ohjausdatan virrat MCU:lle.

2) Kun MCU on hyväksynyt siirron, se tiedottaa tästä USQ-logiikalle, joka edelleen siirtää tiedon siirron pyytäjälle, eli PBX:lle, joka vuorostaan käskee GW:tä lopettamaan muuntamisen (Kuva 4.15)



Kuva 4.15: Siirtopyynnön vahvistaminen ja GW:n neuvonta

3) GW on lopettanut medianmuunnoksen ja PBX on vapaa siirtämään puhelun toiselle PSTN-puhelulle. Kun siirto on valmis, PBX tiedottaa siitä vielä asianomaiselle logiikkakomponentille (Kuva 4.16).

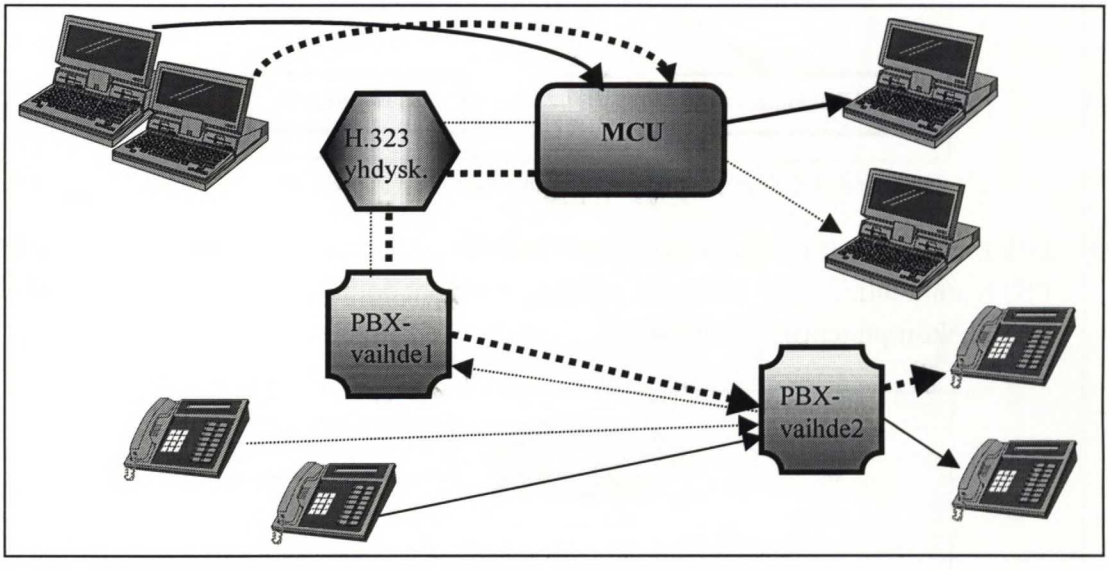


Kuva 4.16: Varsinainen siirto ja siitä tiedottaminen.

4) Yhdyskäytävä voi halutessaan nähdä siirron *hangup*-toimintona, mutta koska se ei sitä notifikaatiota ainakaan logiikkakomponenteille päin voi antaa, ja siihen liittyneet vaihdekomponentit eivät tätä hangup-viestiä huomioi, sillä ei ole mitään

merkitystä. (Tässä yksi syy, miksi logiikka ja yhdyskäytävä olisi mieluummin piilotettava toisiltaan: tietyt vaihteiden keskinäiset operaatiot tulevat helpommiksi, kun logiikkakomponentit puuttuvat niihin mahdollisimman vähän.)

USQ:n tulisi olla mahdollisuuksien mukaan paikkariippumaton. Jos pelkästään sovelluksista koostuvat komponentit on toteutettu jollakin universaalimallilla, kuten Java Beans tai Microsoft COM, voivat kyseiset komponentit sijaita periaatteessa missä päin Internetiä tahansa. Asianlaita on hiukan toinen laitteisto-osia sisältävien komponenttien tapauksessa: vaikka laitteistoja kontrolloivat ohjelmistot voisivat toimintansa puolesta sijaita missäpäin maailmaa tahansa, itse laitteistot saattavat vaatia sellaisia fyysisiä liittymiä, joita ei pitkillä matkoilla voida järkevästi toteuttaa. Esimerkkinä tällaisesta liittymästä on PBX-yhdyskäytävä-liittymä. Jos kuitenkin on tarvetta käyttää yhdyskäytävää muualtakin PSTN-maailmasta, tarvitaan yhdyskäytävän "eteen" yksi ylimääräinen PBX-komponentti, esitetty kuvassa 4.17.



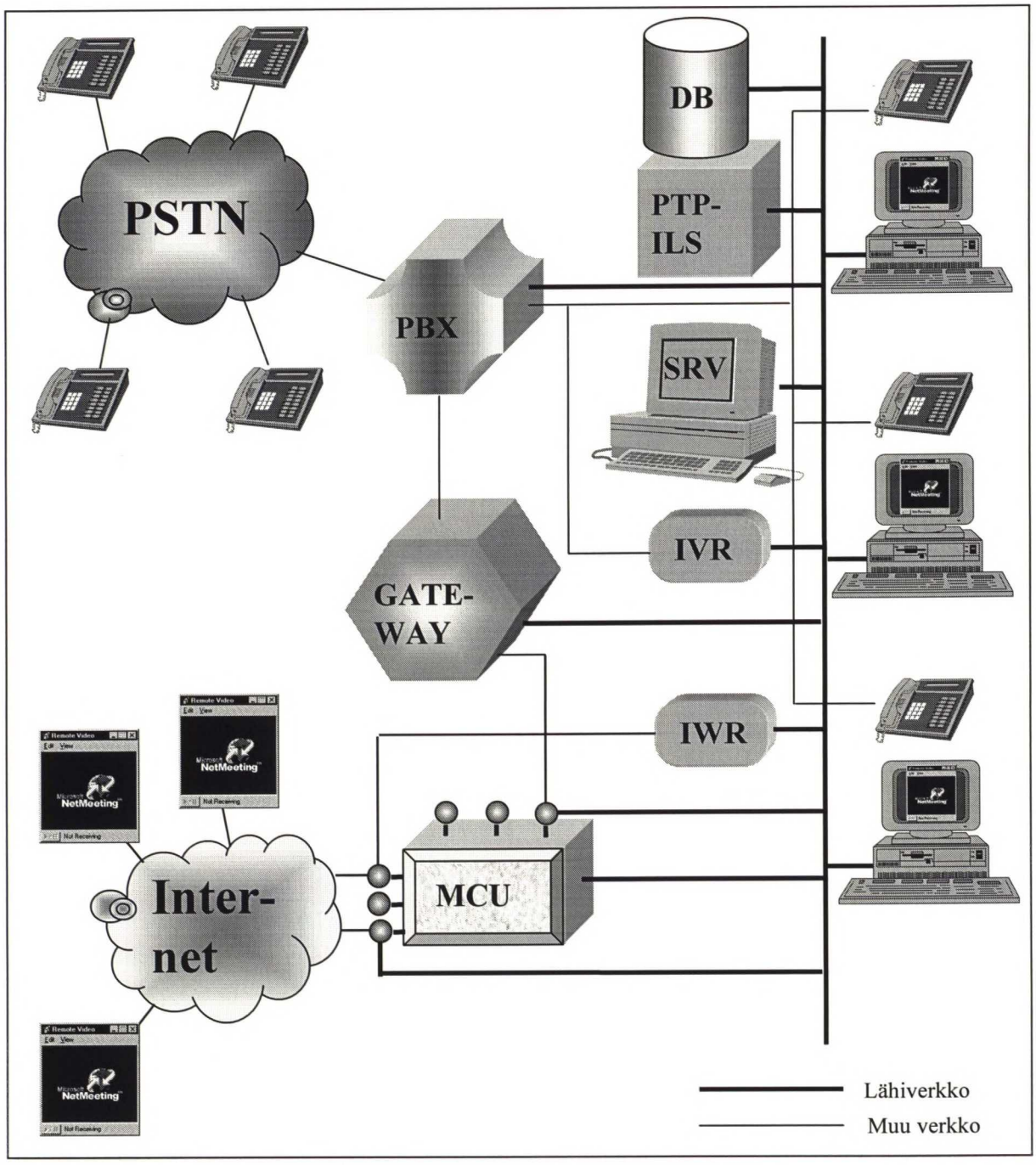
Kuva 4.17: Yhdyskäytävä erillään PBX:stä

Edellä on esitetty useanlaisia eri laitteistoarkkitehtuureja päämääränä maksimaalinen USQ-toiminnallisuus käytettävissä olevalla laitteistolla. Näiden perusteella voidaan todeta, että vaikka puhelunkäsittely onnistuu myös ilman MP:tä ja minimaalisella vaihdetoiminnallisuudella toteutetulla MCU:lla, sekä hyvin yksinkertaisella yhdyskäytävällä, vaatii sekoitettu IP:n ja PSTN:n yhteiskäyttö tehokkaasti ja luotettavasti toteutettuna komponenttina dynaamisesti toimivan GW:n ja MP:n sisältävän MCU:n. MCU:n tulee MP:n lisäksi kyetä toimimaan, kuten usean IP-osoitteen omaava IP-puhelin (kuten IP-vaihde):

- MCU:hun tulee pystyä soittamaan: sen täytyy pystyä valvomaan useita IP-osoitteita
- Kaikkien IP-osoitteiden välillä pitää pystyä ohjaamaan mediavirtaa toisista

riippumatta
- IP-osoitteita tulisi olla useampi samassa fyysisessä koneessa prosessorin, muistin ja väylien asettamissa rajoissa.
- Mediavirtaa pitää pystyä siirtelemään MCU:n IP-osoitteesta toiseen.

MCU:n vaihdetoiminnallisuuden lisäksi myös agentin työasemalaitteilta edellytetään yhtenäistä toiminnallisuutta, so. IP-terminaali pitää pystyä rekisteröimään yhteneväksi puhelunkäsittelylaitteeksi kuin itse puhelinkin. Tätä varten on lisätty PTP-ILS arkkitehtuuriin.



Kuva 4.18: IP/PSTN-USQ-järjestelmän arkkitehtuuri

Jos yhdyskäytävä ja MCU täyttävät edellämainitut ehdot, tulee USQ-järjestelmän arkkitehtuuri näyttämään sellaiselta, kuin se on kuvassa 4.18. Tämä kuva näyttää edelleenkin pelkältä laitteistoarkkitehtuurilta, mutta on itse asiassa sovellusarkkitehtuuri: se ei ota kantaa siihen, missä niiden vaatimat laitteistot sijaitsevat (joskin käytäntö saattaa pakottaa joitakin sovelluskomponentteja laitteiston yhteyteen, kuten esim. IVR:ää ohjaavan sovelluksen). Mukaan on otettu mahdollisimman täydellinen kuvaus USQ-järjestelmän ulkopuolisine lisäosineenkin: agenttien työasemat eivät kuulu määrittelyn piiriin, kuten ei myöskään tietokanta tai ILS (Internet Locator Server, eräänlainen osoitepalvelin), mutta ovat oleellisia järjestelmän käytettävyyden kannalta.

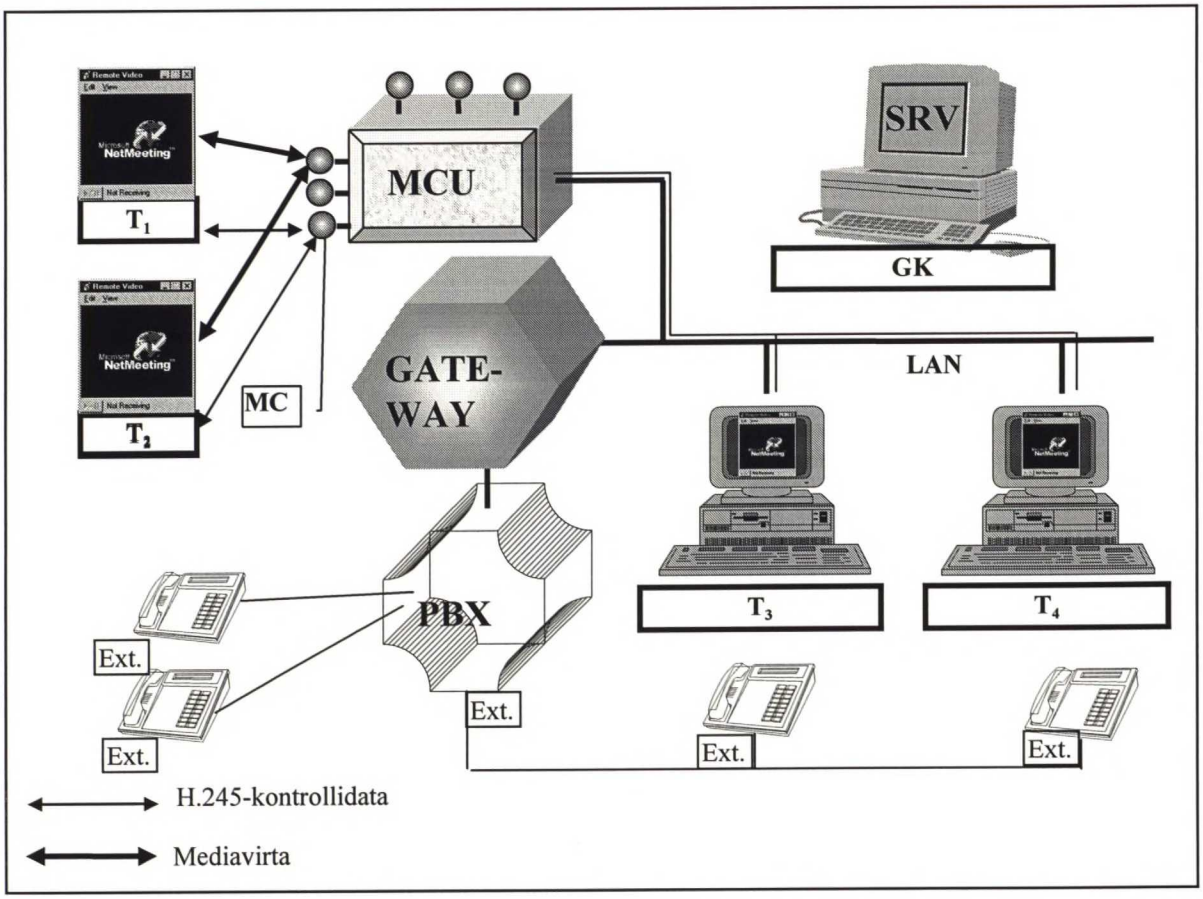
Saapuvan puhelun käsittely tapahtuu, kuten yleisestä arkkitehtuurista kertovassa osiossa on kuvattu, tällä kertaa vaihteina (X_n) toimivat PBX ja MCU, yhdyskäytäväkomponenttina (GW_n) gateway, ja IUR:n virkaa hoitavat IVR ja IWR. Ainoana erona yleisessä arkkitehtuurissa olevaan kuvaukseen on agenttien työpisteiden mahdolliset multimediaominaisuudet (sekä PSTN- että IP-puheluiden käsittelymahdollisuudet), missä tapauksessa yhdyskäytävää ei tarvitse käyttää, vaan valitaan vain agentin käytettävissä olevista medioista se, joka vastaa asiakkaan käyttämää mediaa.

4.2.3 IP-PSTN arkkitehtuuri OSI- CSTA- ja H.323-standardien perspektiivistä

Ylläesitetty arkkitehtuuri ulottuu useammalle kuin yhdelle OSI-pinon tasolle. Suurimmat loogiset erot ovat sovellustason ja esitystapakerroksen välillä, jonka vuoksi on aiheellista esittää systeemi kahdella eri tasolla: OSI-7:n tasolla ja alempien kerrosten näkökulmasta. H.323 käsittelee enimmäkseen OSI-tasoja 4:stä ylöspäin: kuljetuskerroksesta istuntokerroksen ja esitystapakerroksen kautta sovelluskerrokseen [$I_{TUT}96$, $S_{TAL}94$]. Siinä missä kuljetus- ja istuntokerrokseen kuuluvat RAS-kanavassa kulkevat H.225.0-viestit, eräät puhelunsignalointiviestit kuuluvat ennemminkin esitystapa- ja sovelluskerrokseen. Sovelluskerrokseen kuuluvia palveluita H.323:ssa on käsitelty melko vähän: lähinnä gatekeeperin portinvartijan tehtävät kuuluvat näihin [$I_{TUT}96$]. Alempaa tasoa kutsutaan tässä H.323-tasoksi sellaisena kuin esimerkiksi NetMeeting sen toteuttaa. OSI-7 taso on taas ennemminkin CSTA-taso C-domainin palveluineen [$E_{CMA}94$], tai vastaavasti H.323-spesifikaation ensimmäisen version implementoimaton osa. Seuraavassa kuvauksessa H.323-tasolla viitataan OSI-pinon kerroksiin 4-6 ja CSTA-tasolla OSI-7:ään.

H.323-tasolla kuvassa 4.19 esitetyssä systeemissä on neljä erillistä komponenttia osallistumassa varsinaiseen puhelunohjaukseen: asiakaspuolen terminaali, jolla käsitetään PSTN- ja IP-puhelin; vaihdekomponentit, eli MCU ja PBX; yhdyskäytäväkomponentti, gatekeeper ja agenttipään terminaali. Kuvassa 4.19 selvitetään rakenne komponenteittain identifioituna H.323:n kannalta.

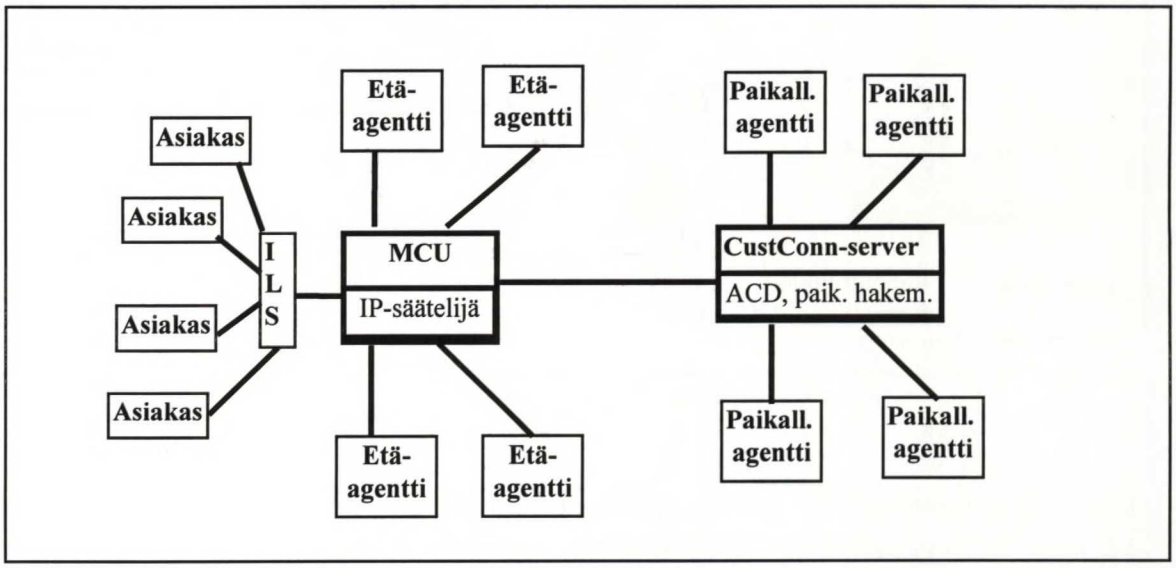
Kuvassa 4.19 käytetyt lyhenteet ovat T terminaalille, MC MCU:n kontrolliosalle (H.245- ja H.225.0-viestien välittäjälle), GK gatekeeperille ja "Ext" ulkopuolisille järjestelmille. Kontrolli- ja mediavirran oletetaan molempien kulkekvan zonea myöten s.e MCU-Terminaali – välillä kulkee media- sekä kontrollivirta. Vastaavasti MCU-yhdyskäytävä (GW)-välillä kulkee ainoastaan mediavirta. Ulkopuolisten järjestelmien informaatiovirtoihin ei tässä oteta kantaa.



Kuva 4.19: Systeemi H.323-tasolla (OSI 4-6)

Gatekeeper-kontrollivirta kulkee tässä RAS-kanavan tavoin erillisenä, ja vaikka käytettävissä onkin MP tai yhdyskäytävä, RAS-viestit menisivät silti irrallisena. Huomattava on, että tässä tasossa ei gatekeeperin kanssa vielä kommunikoida: terminaalit vain ottavat toisiinsa yhteyksiä näennäisen satunnaisella tavalla. Puhelun alustusdata (kuten ominaisuuksien vaihto) ja itse mediavirta kulkevat tällä tasolla, mutta kaikki komponentit ovat vielä irrallisia tai ainoastaan satunnaisesti assosioituja. Gatekeeper on H.323-tasolla irrallaan muusta systeemistä, ja esitetään kuvassa 4.19 ainoastaan ylemmän tason kuvan vertailua varten.

Jos korkeamman abstraktiotason palvelut kuten hakemistopalvelut ja korkeamman tason puhelunohjauspalvelut (ei-triviaali ACD, esim. skill-based routing) ovat tarkastelun kohteena, ylläoleva kuvaus ei enää riitä. Tarvitaan OSI-7-tason kuvaus, jotta voidaan nähdä MC:n, terminaalien ja gatekeeperin välinen kommunikointi. Alla kuvatussa (Kuva 4.20) näkymässä systeemin arkkitehtuurista mukana on paitsi MCU:n ja gatekeeperin välinen yhteys, myös hakemistopalvelin (esim. ILS). Varsinaisessa arkkitehtuurissa ei tällaista ole, koska se on oikeastaan hyvin itsenäinen komponentti, jota voidaan tietyissä tapauksissa käyttää hyväkseen. Kun normaalisti systeemin käyttäjät on jaoteltu joko systeemissä sisällä tai siitä ulkona oleviin, tuo hakemistopalvelin yhden uuden tyypin tähän väliin: ns. etäagentit tai jonkin muun sidosryhmän edustajat [$H_{ANN}96$], joiden intressinä on pysyä näkyvissä, muttei gatekeeperin kannalta kuitenkaan käytettävissä. Kysymyksessä olisivat H.323:n kannalta sellaiset terminaalit, joiden UI:n kautta on määritelty niiden osallistumattomuus gatekeeperin hallinnan alle, jota normaalisti ei voida toteuttaa optionaalisesti. (Normaalitilanne on toistaiseksi se, että terminaali joko on kykenevä gatekeeperin tukemiseen (kuten IPCC) ja rekisteröityy siihen aina, jos ylipäätään haluaa sen palveluja; tai terminaalissa ei ole gatekeeperin tukea. Välimuoto on mahdollinen ainoastaan suunnittelemalla uusi IP-puheluohjelma tai käyttämällä hyväkseen hakemistopalvelun tarjoamia ominaisuuksia.)



Kuva 4.20: OSI-7 tason näkymä systeemistä (kaikki esitetyt yhteydet ovat kontrollisignaaleille: RAS:lle)

Tällä tasolla eri komponenttien funktiot eivät enää ole tismalleen samoja. OSI 4-6-tasoilla terminaaleina tunnetut komponentit ovat nyt agentteja, gatekeeper on muuttunut ACD- ja paikallisia hakemistopalveluja tarjoavaksi komponentiksi, kun taas MCU ei enää olekaan MCU. Sen tehtävinä tällä tasolla on toimia gatekeeperin jatkeena palomuurin ulkopuolella, eräänlaisena IP-säänteljänä ja protokollatulkkina. Sillä on esimerkiksi kyky eritellä siihen

kytkeytyviä web-clientteja näiden IP-osoitteiden tai muiden annettujen tunnisteiden perusteella. Lisäksi se suorittaa tulkkauksen H.323- ja CSTA-maailman (korkean abstraktiotason) käsitteiden välillä.

Yhdyskäytävä- ja vaihdekomponentteja ei enää tällä tasolla näy, koska tässä kysymys on vain agenttien ja asiakkaiden välisestä kommunikoinnista.

Toiminnallisesti katsoen IP-PSTN-järjestelmä toimii sellaisen H.323-systeemin tavoin, jossa gatekeeper on valinnut reitittävän puhelusignaloinnin [$I_{TUT}96$], mutta sallii itse puhelukontrollin (H.245-viestien) kulkea päätepisteeltä toiselle suoraan. Tämä pätee erityisesti jäljempänä esitettävään CustomerConnect 2.0:n IP-yhteyteen: RAS- ja puhelusignalointikanavassa kulkevat viestit reititetään kaikki gatekeeperin kautta, mutta itse mediavirta ja H.245-kontrollidata jätetään kulkemaan pelkästään mediavirran päätepisteiden välille.

5. Case: CustomerConnect 2.0 IP-yhteys

CustomerConnect 2.0 on seuraava versio luvussa 2 esitellylle CustomerConnect 1.5:lle. CustomerConnect 2.0 on rakennettu CustomerConnect 1.5:n pohjalle IP-puheluiden ohjauskyky eräänä tärkeimmistä muutoksista [HM&V98]. Täten puhuttaessa IP-yhteyden "emo-ohjelmistosta" tarkoitetaan joko CustomerConnect 1.5:tä, kun puhutaan kehityksen lähtökohdista, tai CustomerConnect 2.0:sta, kun puhutaan varsinaisen sovelluslogiikan keskusohjelmistosta.

CustomerConnect 2.0 on ensimmäinen versio USQ-määrittelyn toteuttavasta järjestelmästä. Käsitellyinä medioina ovat IP ja PSTN. Seuraava käsittely jakautuu neljään osaan: asetettuihin tavoitteisiin, yksityiskohtaiseen sovellusarkkitehtuuriin, komponenttien tarkempaan selvitykseen menemättä kuitenkaan teknisiin yksityiskohtiin, ja huomioihin jatkokehityksestä. Yhteydet eri standardeihin käyvät ilmi teknisestä osiosta, ja vastaavuudet H.323 – kontrollointikanaviin liitteestä B. Yksityiskohtaiset tekniset selvitykset on esitetty liitteissä A (tekninen dokumentti).

5.1 Tavoitteet

CustomerConnect 2.0:n IP-yhteydelle asetetut tavoitteet yhtenevät koko ohjelmistolle asetettujen tavoitteiden kanssa [HM&V97b]: lopullisena tähtäimenä on USQ-toiminnallisuus siihen liitettävine komponentteineen. Täysin mediariippumattomaan järjestelmään ei vielä tämän työn puitteissa pyritty, vaan tarkoituksena oli aikaansaada yhtenevä käsittely inbound-suuntaisille palvelupyynnöille. Tämä sisältää ohjelmiston sisäiseen arkkitehtuuriin tulevia muutoksia yhtälailla kuin uusien komponenttien konstruoimista, joista ainoastaan viimeksi mainittu kuuluu kokonaisuudessaan tämän dokumentin piiriin – vaikkakin liittymän toteaminen toimivaksi vaati myös eräitä suurehkoja muutoksia itse emo-ohjelmiston infrastruktuuriin.

Tavoite on helpoimmin ilmaistavissa haluttuna toiminnallisuutena: on pystyttävä saamaan aikaan käyttäjien kannalta yhtenevät toiminnot sekä PSTN- että IP-puhelun tekemiseen asiakkaalta agenttiryhmän abstraktion kautta jollekin agentille, sekä lopettamaan tämä puhelu hallitusti. Tässä tärkeimmät kohdat ovat puhelun aikaansaaminen ja lopettaminen CustomerConnect-serverin kontrollin alaisena.

Alla kuvatussa arkkitehtuurissa tämä tarkoittaa nk. Web-Clientin (= IP-CustomerClient = IPCC), MCU:n ja IPX:n (gatekeeperiin eli CustomerConnect-serveriin liitettävä ohjelmakirjasto, joka tukee gatekeeperin sisäistä arkkitehtuuria) sekä niiden välisen liikennöinnin suunnittelua ja implementointia. Kuvaan liittyvät myös käytettävien kolmansien osapuolien ohjelmistojen kontrolloinnin selvittäminen.

Toiminnallisuuden lisäksi tulisi pyrkiä mahdollisimman laajaan alustatukeen, so. konstruoimaan järjestelmä, joka pystyy kommunikoimaan hyvin heterogeenisten osapuol-ten kanssa. Tämä implikoi standardien ja *de facto* – standardirajapintojen hyödyntämistä.

Ensi vaiheessa tärkeintä ja diplomityön alaksi riittävää ovat juuri nämä USQ-funktioihin liittyvät tavoitteet. Näiden lisäksi ohjelmiston teossa tulisi pyrkiä yleiseen tyypilliselle usean käyttäjän sovellukselle tarkoittuihin hyviin ohjelmointitekniikoihin: modulaarisuuteen, tehokkuuteen (monisäikeisyys), luotettavuuteen (synkronointi ja virhetilanteiden käsittely) sekä skaalautuvuuteen. Modulaarisuuten ja skaalautuvuuteen puututaan jo arkkitehtuurivaiheessa, kun taas säikeistykseen vasta ohjelmakomponenttien tarkemmassa esittelyssä (liite A).

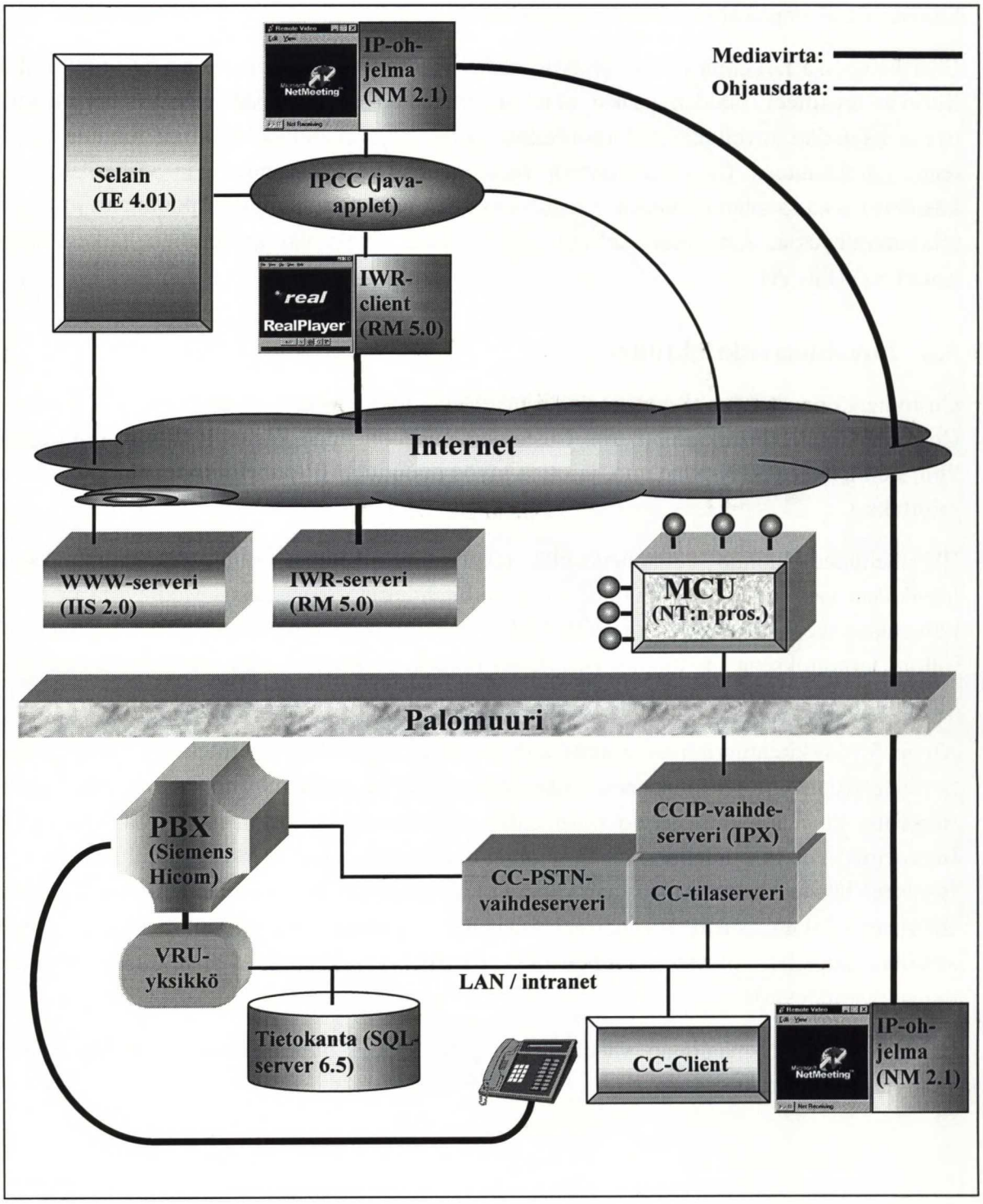
5.2 Sovellusarkkitehtuuri

CustomerConnect 2.0:n IP-yhteys vaatii rakenteen, joka toisaalta tukee H.323:a ja toisaalta CustomerConnectin arkkitehtuuria. CustomerConnectin tulisi käsitellä IP-puheluita aivan kuin ne olisivat PSTN-puheluita, ja ottaa myös huomioon IP-puheluilla toistaiseksi olevat rajoitukset.

IP-puheluiden luonne tämänhetkisellä tekniikalla aiheuttaa muutoksia enimmäkseen logiikkaan sen sijaan, että muuttaisi itse arkkitehtuuria olennaisesti. Mediavirta ei pysty kulkemaan keskitetysti, ja toisaalta kaikki puhelunohjausoperaatiot eivät ole tuettuja, jolloin tarkoituksena on tuottaa keskitetyn ohjauksen illuusio niin arkkitehtuurissa kuin operaatioissakin.

Kuvan 5.1 arkkitehtuurissa on esitetty kokonaiskuva järjestelmästä eroteltuna mediavirtojen ja ohjausdatan virtojen mukaisesti. Itse verkkoyhteydet eivät tule niinkään selvästi esille (tarkempi kuvaus teknisessä dokumentaatiossa liitteessä A), joskin on selvää, että kaikki Internetin samalla puolella sijaitsevat ohjelmistokomponentit (serverit) kuuluvat samaan fyysiseen lähiverkkoon. Ainoastaan välissä oleva palomuurikone erottaa LAN:n intranet- ja ekstranet-osiot toisistaan. Vastaavasti asiakkaan koneessa pyörivät kaikki Internet-pilven toisella puolella olevat komponentit (näistä vain applet ajaa itse selaimen muistiavaruudessa).

H.323:n ja PSTN-maailman vaatimuksien muokkaama hybridiarkkitehtuuri on esitetty yksityiskohtaisesti kuvassa 5.1.



Kuva 5.1: CC-IP 2.0 arkkitehtuuri

Tässä luvussa käsiteltävät osat ovat IPCC (IP Customer Client) ulkopuolisine sovelluksliittymineen, MCU ja IPX (IP-vaihdeserveri). Muiden komponenttien merkityksestä kerrotaan sen verran kuin kokonaisuuden ymmärtämiseksi on tarpeen.

H.323-järjestelmän terminaalina toimiva IPCC, samoin kuin IP-puheluohjelmisto ja IWR-client ovat asiakkaalle näkyviä komponentteja. Näistä kaksi jälkimmäistä ovat Microsoftin NetMeeting 2.1 – ActiveX-kontrolli ja RealMedian RealPlayer 5.0:n ActiveX-kontrolli. Kummatkin kontrollit ovat tyypillisesti asiakkaan järjestelmässä valmiina, mutta ohjauskäskyt latautuvat IPCC:n mukana. IPCC ajaa asiakkaan järjestelmässä pyörivän selaimen muistiavaruudessa, kun selain on ensin hakenut IPCC:n sisältävän web-sivun CustomerConnect-palvelua tarjoavan tahon www-serveriltä.

IPCC:n tehtävänä on toimia asiakaspuolen IP-puhelinkontrollina, vaikka se todellisuudessa toimii *sekä* asiakkaan *että* agentin IP-puhelimien kontrolloijana. Tämän tilanteen syyt ovat puhtaasti tekniset: koska tarkoitukseen sopivaa keskitettyä IP-virran hoitavaa komponenttia ei ollut saatavilla, oli IP-puhelut kontrolloitava viime kädessä komentamalla virran solmukohtia (päätepisteitä, jos kyseessä kahdenvälinen puhelu). Koska yhtenevän käsittelyn vaatimus agenttipäässä estää agentin IP-puhelimen kontrolloinnin, ainoa ja onneksi myös riittävä mahdollisuus on kontrolloida asiakkaan IP-puhelinta. Tämä on riittävä siksi, että IPCC on aina puhelun aloittaja, ja siten myös konferenssin isäntä. Sikäli kun NM-API sen sallisi, olisi täysin mahdollista tehdä jopa CustomerConnect:n määräämä puhelunsiirto pelkästään IPCC:tä komentamalla [MICR97a].

Mediavirta (siis AV) kulkee ainoastaan IP-puhelimien kautta: agenttipäässä ei ole edes IPCC:tä. Vaikka aikaansaadaankin illuusio mediaviran keskitetystä kulkemisesta, niin todellisuudessa järjestelmä ei ota kantaa mediavirran reittiin. (Ajan suhteen vakiona pysyvään reittiin tarvitaan erilaisia protokollia, joita mm. TAPI 3 hyödyntää [MICR97c].)

IPCC:n suhde muihin asiakaspään komponentteihin on joko palvelin, mitä se on selaimeen nähden, koska selain pyytää sitä suorittamaan tiettyjä toimintoja käyttäjän/asiakkaan toivomusten mukaisesti; tai client, mitä se on käyttämiinsä ActiveX-kontrolleihin nähden. Yhteydet on kaikki koodattu HTML-sivuun, joten selain näyttelee yhdessä WWW-serverin kanssa tärkeää osaa tavoitettavuusinformaation jakelussa.

Heti HTML-sivun latauduttua IPCC ottaa yhteyden MCU:hun, joka välittää vain kontrollidataa edelleen gatekeeperinä toimvalle CustomerConnect-serverille. H.323-näkökulmasta tämän järjestelmän MCU on lähes pelkkä MC, mutta vaihdemaailmasta katsoen järjestelmän MCU toimii PBX:n abstraktiona. MCU on tässä tapauksessa C++:lla tehty Windows NT:n prosessi, joka kuuntelee verkon yli tapahtuvia IPCC:den yhteydenottoja ja viestejä. IPCC:n viestit MCU kääntää ja paikkailee vaihdemaailmaan sopiviksi, mutta ei vielä tee täydellistä CSTA-muunnosta.

On luonnollisesti mahdollista asentaa koko H.323-CSTA – käännöstoiminnallisuus samaan moduliin, mutta tämä sotisi modulaarisuutta ja standardisointia vastaan eritoten jatkokehitystä ajatellen. Koska terminaalin on rekisteröidyttävä gatekeeperille, jos sellainen on olemassa, ja koska CustomerConnect kuitenkin ymmärtää hallitsemansa alaliittymät monitoroitavina laitteina [HM&V97d], tapahtuu rekisteröityminen MCU:n kautta niin, että terminaalit (IPCC:t) luulevat rekisteröityvänsä gatekeeperiin suoraan ja gatekeeper kuvittelee vain monitoroivansa hiukan uudentyyppisiä alaliittymiä.

Arkkitehtuurissa MCU on piirretty palomuurin ulkopuolelle tietoliikennöinnin helpottamiseksi terminaalien ja MCU:n välillä. Tämä ei kuitenkaan estä MCU:ta sijaitsemasta myös palomuurin sisäpuolella, mikäli muuri vain on tarpeeksi tarkasti konfiguroitavissa.

MCU:n lävitse ei kulje mediavirtoja, ainoastaan ohjausdataa, vaikka pidetäänkin yllä illuusiota keskitetystä mediavirran solmukohdasta. Tosiasiallisesti MCU on vain kontrollidatavirran solmukohta.

CustomerConnect serveri on jaettu USQ-serverille ominaisesti vaihteesta riippumattomaan tilaserveriin ja vaihteen yhteydessä toimivaan "vaihde-USQ-ajuriin", eli vaihdeserveriin [HM&V97c]. Kukin vaihdeserveri on vaihdekohtainen, ja toimivan yhteyden saamiseksi tilaserveriin jokaisen vaihdekomponentin tulee toteuttaa oma vaihdeserverinsä. IP-vaihteen, l. MCU:n "ajuri" on PBX-vaihdeserverin rinnalla toimiva IP-vaihdeserveri, IPX. IPX tekee lopulliset muutokset H.323-käsitteistöön siten, että tilaserveri näkee pelkän CSTA-mallin. IPX:n tehtäviin kuuluu myös MCU:n käyttämän gatekeeper-pohjaisen tiedon varastointi aivan samoin, kuin IPX voi tarkistaa joitakin H.323-puheluun liittyviä tiloja MCU:lta.

IPX on myös C++:lla tehty moduli, mutta se ei aja itsenäisenä prosessina, vaan muodostuu pikemminkin joukosta kirjastofunktioita ja –objekteja, jotka linkitetään staattisesti käännöksen aikana muuhun CustomerConnect-serveriin.

Arkkitehtuuri on useampikerroksinen: kolme kerrosta muodostuu erilaisista terminaaleista (puhelimet, IP-puheluohjelmat, työsemat) koostuvan käyttäjäkerroksen, vaihteista / vaihdesimulaattoireista koostuvan laitteistokerroksen ja tilaserverin muodostaman logiikkakerroksen yhteenliittymänä. Vertaus tyypilliseen kolmikerrosarkkitehtuuriin ei ole kuitenkaan aivan selkeä: tyypillisesti alempi kerros toimii aina serverinä ylemmälle kerrokselle, mutta tässä tapauksessa PBX on liian arvokas resurssi muodostaakseen alimman kerroksen PSTN-puolelle, kun taas MCU toistaiseksi pelkkänä prosessina on melko suoraviivainen toteutukseltaan tilaserveriin verrattuna, joten IP-maailman silmissä tilaserveri on alimmassa kerroksessa.

Useampikerroksisuus ennakoi hyvää skaalautuvuutta: periaatteessa yhtä CustomerConnectin IP-vaihdeserveriä kohden voi olla monta MCU:ta, ja yhtä MCU:ta kohden taas monta web-clienttiä. Koska MCU:n on melko kompakti, sitä ajavalle tietokoneelle jää paljon vapaata muistia useiden clienttien käsittelyyn. Vaikka CustomerConnectin täytyykin

käsitellä näitä kaikkia clientteja, vaadittujen tietoliikenneyhteyksien viemä muisti vähenee radikaalisti, koska kaikki viestit tulevat saman socketin kautta. Viestejä on tosin huomattavasti tiheämmässä, mutta tämäkin on oikeastaan vain aiemmin käyttämättömäksi jääneen tietoliikennekapasiteetin hyödyntämistä.

Palomuuri yrityksen intranetin ympärillä aiheuttaa erinäisiä hankaluuksia systeemille: lähinnä kyse on IP-osoitteista ja palomuurin läpäisemien mediavirtojen säätelystä. Eräät palomuurit eivät nimittäin tietyllä tavalla konfiguroituna päästä lävitseen kuin datavirran [MICR98]. NetMeetingissä onkin määritelty ne loogiset portit ja protokollat, joita palomuurin on laskettava kulkemaan. IP-osoitteen sisältämään ongelmaan palataan myöhemmin terminaali-komponentin yhteydessä liitteessä A.

Muut järjestelmän komponentit ovat IWR-järjestelmän (Interactive Web Response) [EFUS97] gatekeeperin kanssa samassa lähiverkossa sijaitseva mediavirtapalvelin; niinikään samassa LANissa oleva WWW-serveri; tietokanta sekä VRU-yksikkö. Tietokanta ja VRU-yksikkö eivät ole IP-yhteyden kannalta olennaisia, paitsi mitä tulee IP-puheluiden ja puhelunohjausympäristön ylläpitoon. Sen sijaan asiakkaan selaimelle dataa lähettävillä serverikomponenteilla on isompi merkitys. Molemmat toimivat informaatiovirran tuottajina: WWW-serveri (tässä IIS 2.0) tavoitettavuusinformaatiokanavan lähteenä ja IWR-serveri (tässä RealServer 5.0) jononkäsittelymediavirran (AV) tuottajana.

WWW-serverin rooli USQ:n käsitteistössä on PSTN-maailmaan verrattuna aivan uusi: uuden kontaktipisteen julkituonti ei tyypillisesti hoidu minkään automatiikan avulla, vaan siitä on tiedotettava erikseen muuten. IWR viestittää keskitetysti suuruusluokkaa suuremmalle client-määrälle monipuolisempaa mediatarjontaa kuin mihin IVR kykenee.

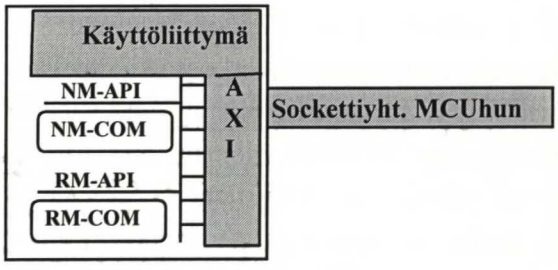
5.3 Tekninen kuvaus komponenteittain

5.3.1 Terminaali

Terminaalina toimii selaimen muistiavaruudessa suorittava java-applet. Tämän IPCC:n tehtävänä on kontrolloida samassa muistiavaruudessa sijaitsevia NetMeeting 2.1 IP-puhelimen ja RealMedia IWR-clientin COM-objektien ActiveX-kontrolleja, sekä kerätä asiakkaalta/käyttäjältä tunnistetietoja välittämällä asiakkaasta kerätty tieto edelleen MCU:n kautta gatekeeperille. Yhteys IPCC:stä CustomerConnect-serveriin on rakennettu HM&V Telecommunicationsin kehittämän alhaisen tason TCP/IP-sockettirajapintaa [HM&V98] hyväksikäyttävinä H.323:n RAS- ja puhelusignalointikanavina gatekeeperin MCU:hun (jolla tässä viitataan palomuurin ulkopuolella olevaan komponenttiin).

IPCC jakautuu kolmeen osaan: ActiveX-kontrollirajapintaan (AXI), käyttöliittymään (UI) ja sockettiyhteys MCU:hun. UI on parametrisoitu niin, että käyttäjälle (asiakkaalle) voidaan

tarvittaessa näyttää hyvinkin erilaisia liittymiä pelkästä napista täysimittaiseksi dialogiksi. AXI sisältää sekä NetMeeting- että RealMedia-kontrollifunktiot, joista NetMeeting-rajapinnassa on sekä kahvat NetMeeting 2.0 SDK:ssa dokumentoituihin COM-objektin tukemiin funktioihin [MICR97b], että viestienkäsittelijät (event-handlers) NetMeetingin tarjoamille viesteille. RealMedia-rajapinta käsittää hyvin rajoitetun valikoiman funktioita ja viestienkäsittelijöitä, lähinnä vain IWR:n käynnistämiseen ja lopettamiseen tarvittavat funktiot, sekä mediavirran loppumisesta/katkaisusta ilmoittavien viestien käsittelyfunktiot. Sockettiyhteyteen sisältyy infra datan lähettämiseen ja vastaanottamiseen verkon ylitse. Rakenne on esitetty kuvassa 5.2.



Kuva 5.2: IPCC:n sisäinen rakenne

IPCC:n tarkoituksena on toteuttaa gatekeeperin tuki NetMeetingin ympärille. Tätä varten sen on ilmoittauduttava gatekeeperille (CustomerConnect serverille) ja välitettävä kaikki puhelunohjaustoiminnot gatekeeperin kautta [ITUT96]. Kun käyttäjä tai NetMeeting aloittaa jonkin tapahtumasarjan, applet huomaa sen ja välittää vastaavan tiedon MCUlle socket-rajapintaa käyttäen. Tämä rajapinta, nimeltään iC-rajapinta, on rakennettu Customer-Connect – serverin tukemien X-ja A-rajapintojen pohjalla olevan infrastruktuurin päälle. Toteutettu viestimekanismi on monisäikeinen ohjelma, jonka synkronointi tapahtuu jonorakenteen avulla [HM&V98].

IPCC käynnistyy automaattisesti, kun asiakas lataa IPCC:n sisältävän sivun selaimeensa. (IPCC voi myös odottaa passiivisena, mikäli se on siten parametrisoitu, mutta tyypillisessä tilanteessa yhteys otetaan heti). Tyypillisen käynnistymisen aloittaa selain, joka luo instanssit kaikista sivulla olevista appleteista ja kutsuu erikseen jokaisen `start()`-metodia. Tämä puolestaan käynnistää itse IPCC-säikeen, joka alustuu kutsumalla MCU:ta ja aloittamalla sockettisäikeen. Yhteys MCU:hun sisältää kirjautumisen ja tarvittaessa gatekeeperin resurssitietojen (ryhmädatan) haun.

Alustuksen jälkeen IPCC ottaa sivuilla olevien NetMeeting- ja RealMedia-COM-komponenttien kontrollin itselleen. Tämä tapahtuu, kun selain kutsuu samalle sivulle kirjoitettua VBScriptin `window_OnLoad()` – metodia, joka antaa kyseisten komponenttien viittaukset IPCC:lle. Saatuaan viittaukset mediavirran käsittelykomponentteihin IPCC alustaa ne käyttövalmiiksi.

Alustava yhteys MCU:hun otetaan synkronisella komennolla. Kun on varmistuttu MCU:n olemassaolosta, sockettisäie asetetaan asynkroniseen kuuntelutilaan molemminsuuntaisen yhteyden ylläpitämiseksi: MCU:n kautta tulevat gatekeeperin viestit parsitaan tässä säikeessä, jonka jälkeen kutsutaan AXIn funktioita. Tätä säiettä käytetään myös puhelutapahtumien jonkinasteiseen synkronointiin: tulivat viestit sitten gatekeeperiltä tai AXIsta, ne kulkevat saman säikeen funktioiden kautta.

Appletin tärkein toiminnallisuus käsittää kolme funktiota: `call`, `hangup` ja `transfer` (odottaa NM-API:n tukea [MICR97a]) Monet Call-Center-toiminnot saadaan aikaiseksi pelkästään viivästetyllä `call`-funktiolla. Esimerkiksi jonotus, kutsunsiirto, puhelun esto ja ryhmäpuhelu (B-yhteyden puolella, ei asiakkaan päässä) ovat kaikki MCU:n ja gatekeeperin yhteispeliä puhelun varsinaisen aloittamisen viivästämisessä. Kun asiakas on esimerkiksi valinnut IPCC:n UI:stä ryhmän ja aloittanut ryhmäpuhelun (joko painamalla ryhmän nappia, tai valitsemalla ryhmän listalta ja painamalla "soita"-nappia) tekee IPCC seuraavaa:

- lähettää **IC_CALL_SERVICE** – viestin MCU:lle
- aloittaa IWR-viestin "soittamisen" (ei sijaitse CustomerConnect-serverin keskitetyssä jonokäsittelyssä)
- tyhjentää UI:n jättäen tilalle vain yhden napin ("Keskeytä jonotus")
- komentaa RealMedia-clientia aloittamaan mainoksensa
- RealMedia-client pyytää RealMedia-serveriltä mediavirtaa määrätyllä tunnuksella
- RealMedia-serveri aloittaa mediavirran lähetyksen
- jos asiakas painaa "Keskeytä jonotus"ta sillä aikaa, kun jonoviesti soi:
  - IPCC lähettää **IC_CALL_HUNGUP**-viestin MCU:lle
  - IPCC seisauttaa RealMedia-clientin
  - IPCC palauttaa aikaisemman UI:n ja palaa aiempaan sisäiseen tilaaansa
- jos RealMedia-serverin mediavirta loppuu ennen jonotuksen päättymistä, RealMedia-client pyytää serveriä aloittamaan uudestaan
- jos MCU lähettää **IC_MAKE_CALL**in ja osoitteen, se tarkoittaa, että ryhmäpuhelulle on löytynyt vastaanottaja, jolloin:
  - vanha UI palautetaan jokainen kenttä ja nappi disabloituina
  - IPCC komentaa NetMeeting-kontrollia aloittamaan mediavirran lähettämisen
- jos MCU lähettää **IC_ERROR**in selityksineen (ei agenttia vapaana / CustomerConnect-serveri eli gatekeeper ei vastaa):
  - vanha UI palautetaan, ainoana erotuksena ryhmälistan tilalla oleva "ylivuodon käsittely", eli viesti, joka pyytää jättämään palautetta.
  - jos tätä painetaan, paikallinen sähköposticlient aukaistaan.

`Hangupin` käsittely on hiukan vaativampaa, koska siinä voi olla kyse kolmenlaisesta viestistä: joko on kyse asynkronisesta viestistä, joka tulee gatekeeperiltä, IPCC:n UI:stä tai NetMeeting:n omasta UI:stä. Operaatiot ovat seuraavanlaisia:

- Jos asiakas painaa HANGUPia NetMeeting:n UIstä:
  - NetMeeting lähettää **MEMBER_CHANGED**-viestin IPCC:n AXIin
  - **MEMBER_CHANGED** jää sivulla olevien VBScript-viestinkäsittelijöille
  - VBScript-viestinkäsittelijä antaa viestin appletille
  - applet päättelee viestin johtuvan paikallisesta puhelunkatkaisusta ja lähettää **IC_CALL_HUNGUP**in MCU:lle.

- IPCC enabloi koko käyttöliittymän ja palaa normaaliin sisäiseen tilaansa.
- Jos agentti painaa HANGUPia NetMeeting:n UI:stä,
- Asiakkaan NetMeeting hoksaa, että eräs jäsen juuri poistui konferenssista.
- NetMeeting lähettää **MEMBER_CHANGED**-viestin IPCC:n AXIin
- **MEMBER_CHANGED** jää sivulla olevien VBScript-viestinkäsittelijöille
- VBScript-viestinkäsittelijä antaa viestin appletille
- appletti päättelee viestin johtuvan etäpuhelimen katkaisusta, joten mitään ei toistaiseksi tehdä.
- Tämä konferenssi koostui ainoastaan kahdesta osanottajasta, joten jos toinen osapuoli lähtee konferenssista, se lopettaa koko konferenssin. Siksi NetMeeting lähettää vielä toisen **MEMBER_CHANGED**-viestin
- **MEMBER_CHANGED** jää sivulla olevien VBScript-viestinkäsittelijöille
- VBScript-viestinkäsittelijä antaa viestin appletille
- applet päättelee viestin johtuvan paikallisesta katkaisusta ja lähettää **IC_CALL_HUNGUP**in MCU:lle.
- IPCC enabloi koko käyttöliittymän ja palaa normaaliin sisäiseen tilaansa.
- jos agentti painaa HANGUPia CustomerConnect-clientin omasta UI:sta
- **X_CALL_HUNGUP** joutuu IPX:n ja MCU:n käsiteltäväksi ja muokkaamaksi, kunnes viimein IPCC:n sockettiyhteyttä kuunteleva säie saa **IC_HANGUP_CALL**-viestin.
- **IC_HANGUP_CALL**-viestin process()-metodi kutsuu AXIn `hangup_call()`-metodia, joka lopettaa konferenssin.
- IPCC enabloi koko käyttöliittymän ja palaa normaaliin sisäiseen tilaansa.

`Transfer` ei itse asiassa ole varsinainen siirto, vaan pikemminkin sarja konferenssiin kutsumisia ja poistamisia. Koska NetMeeting-kontrolli ei tätä tue [MICR97a] (vaikka UI kylläkin tukee) ei tätä tueta myöskään IPCC:ssä. Kun mediavirtakomponentti alkaa tukemaan tätä toimintoa, on toiminta suunnilleen seuraavanlainen:

- jos agentti A käskee siirron itsensä ja asiakkaan B väliltä agentin C ja asiakkaan B väliseksi:
- CustomerConnect-client lähettää gatekeeperin kautta siirtokäskyn IPCC:lle IC-viestinä
- IPCC kutsuu C:n mukaan konferenssiin
- IPCC komentaa NetMeetingin kontrollia siirtämään mediavirrat väliltä A-B välille B-C.
- IPCC komentaa NetMeetingin kontrollia pudottamaan A:n pois konferenssista.

Ylläesitetty tapaus on toistaiseksi ainoa toimiva: jos siirto tarkoittaa sitä, että siirtäjä jättäytyy pois mediavirrasta, ei siirtäjä voi olla konferenssin alkuperäinen isäntä, sillä silloin koko konferenssi päättyy. Lisäksi agenttipään NetMeeting-kontrollia ei voi komentaa, koska sille ei ole omaa kontrollikomponenttia CustomerConnect:n puolella.

Koska käytössä oleva alustalle parhaiten sopiva olioarkkitehtuuri oli juuri COM, ei edes java-appletin rakentaminen tämän ympärille tee IPCC:stä kovin alustariippumatonta: MS:n tietoturvanäkemys pakottaa IPCC:tä käyttävän selaimen olemaan toistaiseksi Internet Explorer 4.0 tai uudempi [MICR98], ja muut kuin Windows-alustat ovat toistaiseksi testaamattomia. Tämä on ongelma yleiskättöisyyttä ajatellen, mutta vain toistaiseksi: käytetyt mediankäsittelykomponentit ovat standardilähtöisiä ja markkinaosuuksiltaan suurimpia kyseiseen käyttöön tarkoitettuja ohjelmistoja [PULV98]; IPCC:n isäntänä toimivan selaimen merkkikään ei rajoita *IPCC*:n alustaksi Windows-alustaa, joskin se on ainoa testattu alusta muiden komponenttien tapauksissa.

Tarkempi dokumentaatio rajapintoineen ja asennusohjeineen on esitetty liitteessä A.

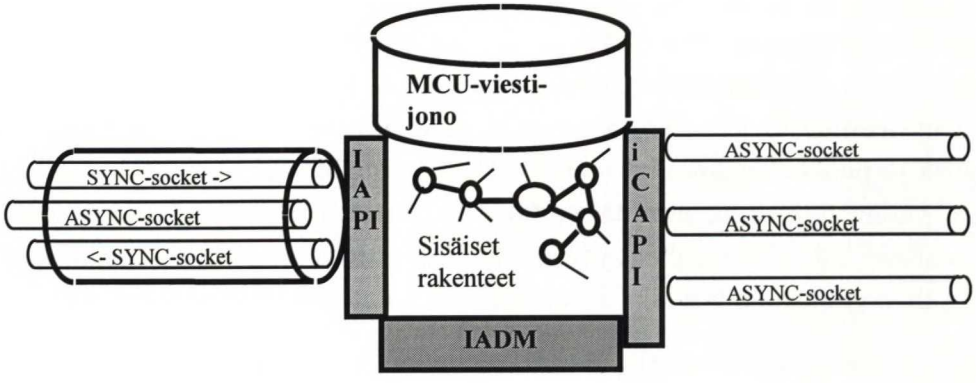
5.3.2 MCU

MCU:lla on hyvin keskeinen merkitys: sen on käyttäydyttävä terminaaliin nähden H.323-mukaisena MCU:na, koska terminaali on pelkästään H.323-riippuvainen [MICR97b]. Sen sijaan gatekeeperin, joka taas tässä toteutuksessa on CSTA-mallin toteuttava komponentti [HM&V98, HM&V97b], on kuviteltava MCU:n olevan jokin S-domainin alidomain, eli vaihde. H.323-perspektiivistä MCU-komponentti on gatekeeperin yhteydessä oleva MCU, sillä sen toiminta on loppujen lopuksi hyvin riippuvaista gatekeeperin tarjoamista palveluista. Ilman MP:tä MCU:lla on suhteellisen vähän sellaista tarjottavaa, mihin terminaalit eivät jo pystyisi. CSTA-mallin näkökannalta MCU on vastaavasti vaihdetta ajavan ohjelmiston pohjana oleva fyysinen vaihde. Tämä näkökanta valittiin jatkokehityksen (keskitetyn middleware-MP:n ilmestyminen markkinoille) ja muun yhteensopivuuden kannalta: H.323 ja CSTA on joko silloitettava tai toisesta luovuttava. CSTA sopii hyvin gatekeeperin sisäiseen arkkitehtuuriin, ja H.323 IP-puheluiden arkkitehtuuriin [ECMA94, MICR97b, HM&V97b], joten kummastakaan ei voi luopua. Oli tehtävä silta näiden maailmojen välille, eli käytännössä konstruoitava jonkinlainen "kääntäjä". Tämän komponentin sijainti oli pitkään kysymysmerkkinä, kunnes parhaimmaksi ratkaisuksi osoittautui "käännöstyön" hajautus usealle komponentille: terminaali pyrkii lähettämään NetMeetingistä jo sellaisia viestejä, että niille on olemassa yksikäsitteinen funktio RAS:stä ja puhelusignalointiviesteistä CSTA-viesteihin (X-rajapinta); MCU lisää "käännökseen" vaihteen (S-domainin funktioita) ja varsinainen liityntä gatekeeperiin (IPX) loput CSTA-mallista, eli vaihde-puhelu-yhteys struktuurit. Koska MCU ei varsinaisesti ole oikea vaihde, siihen viitataan myös aika-ajoin "virtuaalivaihteena".

MCU:n tehtävä on siis suorittaa osa H.323–CSTA-muunnoksesta, mikä tarkoittaa, että sillä on oltava paitsi yhteydet web-clientteihin, myös yhteys itse CustomerConnect-serveriin. Terminaaleihin päin yhteys on joukko iC-rajapinnan kautta keskustelevia asynkronisia socketteja, mutta gatekeeperin puolella on kahden tyyppisiä viestikanavia: synkronisia ja asynkronisia. Normaalisti pelkkä asynkroninenkin kanava riittäisi, mutta on viestejä joihin tarvitaan vastaus kontrollisäikeen suorituksen aikana. Kaikkia tällaisia viestejä taas ei ole toteutettu synkronisina CustomerConnect:n puolella, joten tämä mekanismi tekee taas yhden käännöksen CSTA:n ja H.323:n välillä. Rajapinta gatekeeperiin päin on nimetty I-rajapinnaksi. Tämä rajapinta on jo hyvin lähellä CSTA:ta, noudattaen kuitenkin samaa abstrahointimenettelyä kuten muukin CustomerConnect, eli tunnistaen vain laitteet ja yhteydet. MCU-virtuaalivaihde toimii kuten oikeakin vaihde siinä mielessä, että sille eivät korkean abstraktiotason asiat merkitse mitään: se välittää vain sille ilmoittautuneista terminaaleista sekä näiden tilasta (Connected /Disconnected) ja niihin liittyvistä yhteyksistä. I-rajapinta koostuu lähinnä pyynnöistä (request) ja niiden vastauksista (acknowledgement/confirmation) jotka voivat olla joko hyväksyviä, milloin pyyntö on pystytty

toteuttamaan; tai kielteisiä, jos pyynnön toteuttaminen on osoittautunut mahdottomaksi (kuten olemattoman puhelun lopettaminen).

Edellämainittujen rajapintojen lisäksi MCU:ssa on oltava joukko siihen liityneiden IPCC:n tiloista kertovia sisäisiä rakenteita, koska sen on pysyttävä paremmin ajan tasalla todellisesta tilanteesta kuin CustomerConnect-serverin: jos laitteen tai yhteyden tila muuttuu, MCU:n on tiedettävä se ensiksi. Tästä syystä olemassaolevat yhteydet ja laitteet samoin kuin näihin liittyvä muu tieto on kuvattava MCU:n muistiavaruudessa. Muun muassa gatekeeper-liittymä kommunikoi paljolti tämän tiedon varassa MCU:n kanssa. Pitääkseen sisälle virtaavat komennot ja NetMeetingin tapahtumaviestit järjestyksessä MCU pitää yllä samantapaista, mutta monimutkaisempaa synkronointirakennetta kuin muutkin tämän monisäikeisen ohjelmiston komponentit: viestijonoa. Viestijonon alhaisen tason tehtäviin kuuluu mm. verkon kuuntelu (viestien vastaanottaminen) ja verkkoon kirjoittaminen (viestien lähetys). Korkeammalla abstraktiotasolla viestijono hoitaa useiden lähes yhtäaikaa saapuvien viestin oikean järjestelyn yksinkertaisen jonologiikan mukaan. MCU:n rakenne on kuvattuna kuvassa 5.3

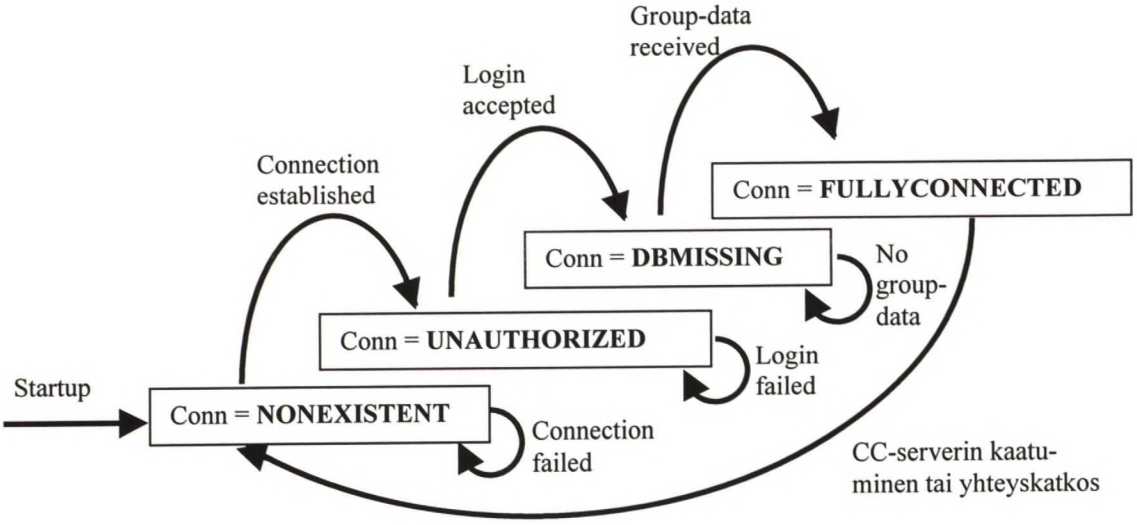


Kuva 5.3: MCU:n rakenne

Kuvassa oleva IADM-rajapinta on pienimuotoinen käyttöliittymä MCU:n testausta ja ylläpitoa varten. (Tämä tulee Windows-sovelluksessa lähes aina "kaupan päälle", jos kyseessä on itsenäinen prosessi, joka ei suorita NT:n palveluna.)

Kun MCU käynnistetään, se luo ensiksi viestijonon ja alustaa globaalit listansa. Alustuksen jälkeen MCU kytkeytyy CustomerConnectin IPX-serveriin. Kytkentäproseduuri koostuu neljästä kytkentätasosta, joista kukin kuvaa CustomerConnect-serverin statusta. Nämä kytkentätasot auttavat MCU:ta paremmin toipumaan erilaisista CustomerConnectin virhetiloista, sekä tiedottamaan näistä kytkeytyneille IPCC:lle (estetään IPCC:n kytkeytyminen silloin, kun CustomerConnect-serveri on vasta käynnistymässä) Kytkentätasot järjestyksessä ovat: **NONEXISTENT, UNAUTHORIZED, DBMISSING** and **FULLYCON-**

NECTED. Aluksi kytkentätaso on **NONEXISTENT**. IPX yrittää kytkeä viestijonoa alustustiedoissa mainittuun osoitteeseen. Jos tämä ei onnistu, MCU odottaa hiukan ja yrittää uudelleen. Jos MCU:n kytkeytyminen IPX:ään onnistuu, se jatkaa seuraavalle tasolle toimiaksen samalla tavoin siellä. Tämä toistuu, kunnes on saavutettu korkein kytkentätaso (**FULLYCONNECTED**). Kuvassa 5.4 oleva graafi kertoo enemmän logiikasta.



Kuva 5.4: MCU-IPX kytkentäproseduuri

Taulukko 5.1: MCU-IPX-kytkentäproseduurin tilat

Taso	Tilaan saapumiseen liittyvät viestit tai funktiot	Selite	Kommentteja
NONEXISTENT	CWinApp::InitInstance(), **Z_CLIENT_DISCONNECTED**	Tällä tasolla IPX:ään ei ole olemassa yhteyttä.	IPCC:t voivat jo kytkeytyä, mutta ainoastaan demotoiminallisuus voidaan taata.
UNAUTHORIZED	CMessageQueue::CreateConnectionTo()	Yhteys IPX:ään on jo olemassa, mutta GK:n resurssidata ja sisäänkirjautumien puuttuvat vielä.	-

DBMISSING	**IS_LOGIN, IS_LOGIN_ACK**	IPX on vahvistanut kirjautumisen, mutta resurssidata puuttuu vielä.	-
FULLYCON-NECTED	**IS_FETCH_GROUPS, IS_FETCH_GROUPS_ACK**	Täysin operationaalinen taso.	Vain tällä tasolla MCU:hun kytkeytyneet IPCC:t saavat varsinaista palvelua.

MCU:n osuus sisääntulevan puhelun käsittelyssä alkaa, kun se saa **IC_CALL_SERVICE** – viestin IPCC:ltä.

- Jos IPX-kytkentätila on **FULLY_CONNECTED**, ja demo-asetuksia ei ole päällä:
  - CMsgICCallService::Process()-metodissa MCU hakee IPCC:n ilmoittaman ryhmänumeron mukaisen ryhmän loogisen IP-numeron.
  - MCU konstruoi **IS_INCOMING_CALL** -viestin, jonka laitetunnisteena (B-numerona) on tavoitellun ryhmän looginen IP-osoite.
  - MCU lähettää **IS_INCOMING_CALL**-viestin asynkronisesti IPX:lle.
  - Jos ja kun IPX lähettää MCUlle päin synkronisessa kanavassa **IS_DIVERT_CALL**in, MCU luo **IC_MAKE_CALL**-viestin **IS_DIVERT_CALL**in parametrien perusteella ja lähettää sen kohteena olevalle IPCC:lle.
  - Koska **IS_DIVERT_CALL** tuli MCUlle päin synkronisessa kanavassa, siihen täytyy vastata samaa kanavaa myöten **IS_DIVERT_CALL_ACK**-viestillä.
- Jos IPX-kytkentätaso ei ole **FULLY_CONNECTED** ja demo-astetuksia ei ole päällä, MCU vastaa kaikkiin IPCC:n pyyntöihin **IC_ERROR**-viestillä.

- Jos demoasetukset ovat päällä, MCU vastaa tällöin kytkentätasosta riippumatta yhteyttä pyytävälle asiakkaalle staattisesti konfiguroitavalla FakeAgentIP-parametrin arvolla: MCU konstruoi **IC_MAKE_CALL**in tällä demo-osoitteella ja lähettää sen yhteydenottopyynnön tehneelle IPCC:lle.

Puhelun katkaisuviesti voi saapua MCU:n kannalta kahdesta paikasta: joko IPX:ltä **IS_HANGUP_CALL**in muodossa tai sitten IPCC:ltä **IC_CALL_HUNGUP**in muodossa. Edellinen tarkoittaa ohjelmallista, CustomerConnectin määräämää puhelunkatkaisua, ja jälkimmäinen manuaalista katkaisua. Jälkimmäisessä tapauksessa MCU yksinkertaisesti konstruoi **IS_CALL_HUNGUP**-viestin ja lähettää sen IPX-serverille asynkronista kanavaa pitkin. Mikäli MCU joutuu käsittelemään **IS_HANGUP_CALL**ia, operaatiot ovat melko lailla samanlaisia kuin MCU:n saadessa **IS_DIVERT_CALL**in IPX:ltä:

- MCU hakee katkaisuviestin parametrina olevan laite-ID:n avulla oikean IPCC:n, jolle katkaisukomento välitetään.
- Jos MCU löytää IPCC:n (asiakas ei ole esimerkiksi lopettanut IPCC:n sisältävän sivun selaamista tai katkaissut puhelua itse), MCU
  - konstruoi **IC_HANGUP_CALL**-viestin

- lähettää **IC_HANGUP_CALL**in löydetylle IPCC:lle
- lähettää **IS_HANGUP_CALL_ACK**in IPX:lle synkronista kanavaa pitkin (koska myös käsky tuli tätä kanavaa pitkin)
- Jos MCU ei löydä IPCC:tä, se lähettää IPX:lle synkronisen kanavan lävitse **IS_HANGUP_CALL_ACK**in toiminnon epäonnistumisesta kertovan negatiivisen paluuarvon kera.

Siirtoa ei MCU:n IPX-rajapintoihin ole lisätty, joten sen toiminnallisuutta ei käsitellä tässä. MCU:n ja IPX:n välisessä rajapinnassa on viesti myös puhelun ohjelmalliseen aloittamiseen, mutta tämä ei koske IPCC:tä, koska se on nimenomaan asiakkaan prosessi, eikä voida näinollen olla varmoja sen tilasta tai edes olemassaolosta. Kyseinen viesti on lähinnä tulevaisuudessa käyttöönotettava laajennus, jota käytetään agentin työasemassa ajavan IP-puhelimen kontrollerin ohjaamiseen.

Tarkempi tekninen kuvaus viestijonodokumentteineen ja rajapintamäärittelyineen löytyy liitteestä A.

5.3.3 Liityntä Gatekeeperiin (IPX)

Varsinaiseen gatekeeperiin terminaali liittyy MCU:n kautta liikennöidessään nk. vaihdeserverin kautta, joka suorittaa lopullisen tulkinnan H.323:sta CSTA:han. CustomerConnect-serverin tilaserveri-osa näkee vaihdeserverin yleisenä CSTA-mallin toteuttavana vaihteen abstraktiona, joka toteuttaa samat käskyt kuin PBX:stä huolehtiva vaihdeserverikin.

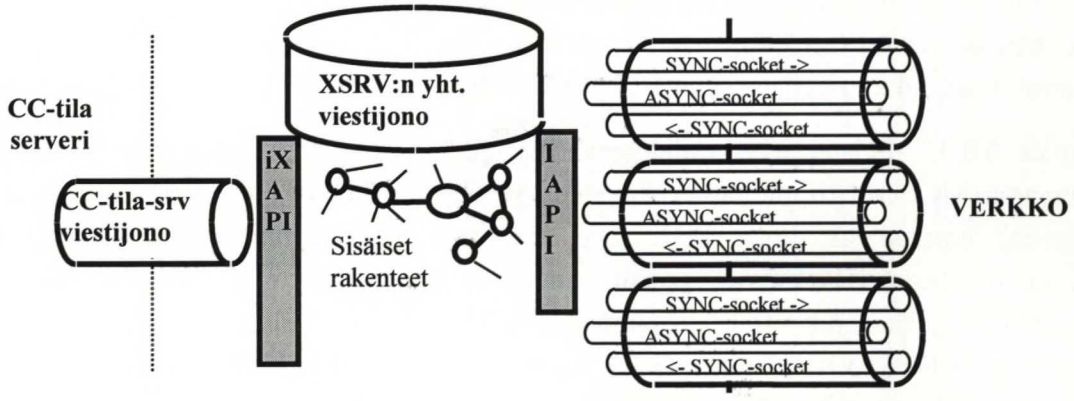
Koska MCU rekisteröityy vaihdeserverille eikä toisinpäin, vaihdeserverin on oltava olemassa MCU:sta riippumatta. (Tämä on perustavaa laatua oleva ero tavalliseen vaihteeseen: normaalisti ilman PBX:ää ei suostuta edes käynnistämään koko puhelunohjaussovellusta [HM&V98]) Toisin sanoen IP-vaihdeserveri ei tee aloitetta MCU:n löytämiseksi, vaan pikemminkin päinvastoin: kun MCU rekisteröityy I-rajapinnan login-viestillä, vaihdeserveri joko hyväksyy loginin ja tarjoutuu jatkossakin välittämään viestit eteenpäin tilaserverille asti, tai kieltäytyy ja pudottaa MCU:n yhteyden pois muistiavaruudestaan. Vaihdekomponentin olemassaolon optionaalisuus ei ensituntumasta huolimatta aiheuta ongelmia, koska jos MCU:ta ei ole olemassa, yksikään IPCC ei pääse kytkeytymään IPX:ään, mikä eliminoi sekä inbound- että outbound-puhelut, sikäli kun ne koskevat CT-järjestelmää. (Jos agentti tekee outbound-puhelun käyttäen jotakin muuta kuin IPCC:tä, esimerkiksi pelkkää NetMeetingiä, kyseisen puhelun oletetaan olevan yksityinen puhelu CustomerConnectin hallinnan ulkopuolella [HM&V98]). Toisaalta MCU ei saa katkaista yhteyttä niin kauan kuin yksikään IPCC on siihen yhteydessä. (MCU:ssa ainoa tapa ottaa MCU irti IPX:stä on lopettaa koko MCU:n prosessi, jota ei tapahdu kuin virhe- ja huoltotilanteissa.)

Tavallisimmat puhelunohjausviestit saavat IP-vaihdeserveriltä CSTA:n mukaisen palvelun, ja ne siirretään riisutun CSTA-mallin tapaan laite-yhteys struktuureina I-rajapinnan lävitse

MCU:lle. Sen sijaan eräistä palvelupyynnöistä ei välitetä eikä lähetetä eteenpäin, koska niille ei ole määritelty mitään merkitystä IP-maailmassa tai niiden toteutus ei toistaiseksi toimi tai ole tuettu [MICR97a].

Ensimmäisestä kategoriasta on esimerkkinä useampi perättäinen kutsunsiirto: koska puhelu on virtuaalinen, kunnes lopullinen määränpää on selvinnyt (mikä taas PSTN-maailmassa voi olla hyvinkin pitkän siirtotien takana) ei ole mitään mieltä tehdä kutsunsiirroille yhtään mitään. Jälkimmäisiä (nollatoteutus) ovat esimerkiksi puhelun siirto ja pito, näiden ollessa oikeastaan saman asian kaksi eri nimeä. (Puhelunsiirto konferensointiympäristössä tehdään kutsumalla uusi jäsen konferenssiin, mutta pitämällä se aluksi "pidossa", eli mediakanavat suljettuina. Tämän jälkeen uusi jäsen otetaan pois "pidosta", eli aukaistaan mediakanavat, kun taas vanha jäsen laitetaan ensin pitoon ja sitten poistetaan konferenssista. Valitettavasti vain mediavirran siirtoa konferenssin osapuolelta toiselle, tai ylipäätään käynnissä olevan mediavirran suoraa kontrollia ei ole käytetyssä IP-puheluohjelman API:ssa tuettu [MICR97a])

IPX:stä voidaan, kuten MCU:stakin, tunnistaa neljä eri osaa. Kuva 5.5 selventää rakennetta.



Kuva 5.5: IPX:n rakenne

Kuvasta voidaan lukea tilaserverin vastainen rajapinta, joka on nimetty iXAPI:ksi; MCU:n vastainen rajapinta, IAPI; monisäikeisyyden synkronoiva viestijono ja sisäiset rakenteet. Viestipohjaiset rajapinnat, sisäiset rakenteet ja viestijonon toiminta on dokumentoitu tarkemmin liitteessä A. MCU:hun päin olevista yhteyksistä on huomattava, että jokainen yhteys koostuu kolmesta socketista: asynkronisesta, ja kumpaankiin suuntaan synkronisesta socketista. Kolme erillistä fyysistä yhteyttä ei siis tarkoita kuin yhtä loogista yhteyttä. Tämän rakenteen syyt on kerrottu tarkemmin liitteessä A. Yhteys tilaserveriin päin on IPX:n näkökannalta aina asynkroninen, koska vaihdekomponentilta saattaa milloin tahansa tulla puhelunohjausviestejä ja toisaalta vaihdeserveri ei näe tilaserveriä serverin ominaisuudessa, joten IPX ei esitä synkronisia pyyntöjä tilaserverille. Tilaserverin kannalta

suurin osa pyynnöistä on synkronisia, mutta koska kyseessä on samassa muistiavaruudessa oleva olio, synkronisia/asynkronisia socketteja ei erikseen tarvita.

IPX ei sisällä mitään pääkäyttäjän UI:tä MCU:n tapaan, koska IPX on osa serveriprosessia. (Hallinnointi tapahtuu tilaserverin pääkäyttäjän käyttöliittymän kautta)

IPX on toteutettu staattisesti linkitettynä kirjastona, joten se suorittaa samassa muistiavaruudessa tilaserverin ja PBX-vaihdeserverin kanssa. Tästä syystä IPX:n ja PBX-vaihdeserverin viestijonot ovat itse asiassa yksi ja sama viestijono. Erotuksena pelkän PBX-serverin tapaukseen USQ-vaihdeserverien viestijono kuuntelee myös itsenäisesti verkkoa TCP/IP-protokollan päällä saapuvien viestien varalta.

IPX-vaihdeserveri käsittelee vain sille itselleen kuuluvia viestejä ja CSTA-rakenteita. CustomerConnectin arkkitehtuurin mukaisesti tila- ja vaihdeserverit ovat itsenäisiä kokonaisuuksia siten, että tilaserverissä on kokonaiskuva CSTA-mallista, kun taas vaihdeserverit pitävät tästä mallista kukin oman mediansa mukaista kopiota [HM&V97a, HM&V98]: IPX:llä on kopiot IP-puheluita koskevista CSTA-rakenteista, ja vastaavasti PBX-vaihdeserveri pitää hallussaan kopioita PSTN-puheluita koskevista CSTA-rakenteista. Tämä taas ei tarkoita sitä, että tilaserverin tiedossa olisi koko ajan puheluiden media; päinvastoin – tilaserverin puhelunkäsittelylogiikka välittää ainoastaan siitä, että jokaisella CSTA-laitteella on yksikäsitteinen ID (tarjontalogiikkalle, so. agenttiryhmälle tarjottujen palvelupyyntöjen järjestykselle mediatyypillä taas on suuri merkitys) . Tämän ID:n perusteella tietoliikennekerros sitten päättelee, minkätyyppisestä mediasta on kysymys – nk. vaihde-ID, eli SWID on liitetty itse laite-ID:n eteen – ja lähettää *jollekin* vaihdeserverille tarkoitetun viestin SWID:n perusteella *kyseistä mediaa* käsittelevälle vaihdeserverille. Tämä on selostettu takemmin liitteessä A.

IPX käsittelee suurta osaa tilaserverin näkemistä laite-ID:stä virtuaalisina. Kun PBX-vaihdeserverissä kaikkia laitteita vastaa jokin fyysinen numerolla identifioitava olio, kuten alaliittymä tai muistipaikka vaihteessa, IPX-vaihdeserverissä ainoita todellisia osoitteita ovat vain asiakkaan ja agentin työaseman IP-osoitteet. Muut osoitteet, kuten IWR- ryhmä ja jonotusosoitteet, joita tilaserveri käyttää toiminnan yhdenmukaistamiseksi, ovat kaikki virtuaalisia. Jopa itse puheluobjektikin on virtuaalinen siihen saakka, kunnes mediavirta todella avataan.

Tässä toteutuksessa IP- ja PSTN-maailma leikkaavat vaihdekomponenttien tasolla vain hyvin vähän. IP/PSTN-yhdyskäytävien aiheuttamaa problematiikkaa ei ole tämän työn puitteissa vielä käsitelty "vaihteiden" vaatimalla tunnistuksen tasolla: vaikkakin media on kätketty tilaserveriltä, olisi tietoliikennekerroksen silti tiedettävä, mille vaihdeserverille lähettää "mixed-media"-puhelunohjauskäskyt (periaate noudattelee luvussa 4 hahmoteltuja linjauksia: media määritellään ohjaustoimintoa pyytävän osapuolen hallussa olevan laitteen mediatyypin perusteella. Tähän ei tosin riitä pelkkä passiivinen SWID-tutkintaan perustuva

logiikka, vaan vaaditaan monimutkaisempaa päättelyä, jonka avulla pystytään selvittämään ohjauspyynnön lähde ja siihen assosioitava laite).

IPX käynnistetään muiden serverikomponenttien mukana CustomerConnectin käynnistyksen yhteydessä. Tilaserveri käynnistyessään alustaa IPX-serverin, joka ensimmäiseksi aloittaa verkon kuuntelun. Tämä on välttämätöntä MCU:n löytämiseksi, jotta CustomerConnectin IP-puoli pääsisi operatiiviseen statukseen. MCU:n ei tarvitse välttämättä ilmoittautua PSTN-toiminnallisuuden alkamiseksi, PSTN:ään liittyvä toiminta eriytyy heti alussa ja eri mediat toimivat jokseenkin itsenäisesti. Aloitettuaan verkon kuuntelun IPX jatkaa laitteiden monitoroinnin (tiettyjen laitteiden/terminaalien valvonnan) käynnistämisellä. Tällä tarkoitetaan agenttien yhteydessä olevien laitteiden tietojen lataamista muistiin, ja monitoroinnin ilmoittamisesta MCU:lle (jossa tästä on melkein nollatoteutus: MCU ei tarvitse kuin tiedot agenttien käyttämistä laitenumeroista / työasemien IP-osoitteista.)

Sisääntulevasta ryhmäpuhelusta IPX saa tiedon MCU:lta viestin **IS_INCOMING_CALL**-yhteydessä. **IS_INCOMING_CALL** sisältää yhden ryhmän virtuaali-IP-osoitteen (viiden kolminumeroisen desimaalisen luvun yhdistelmä, joista neljä ensimmäistä saavat arvoja, joita yksikään binäärinen oktetti ei voi saada), jossa puhelu on – virtuaalisesti – hälyttämässä. IPX jatkaa kuvitelmaa lähettämällä tilaserverille **X_ALERTING** viestin, johon tilaserveri reagoi aloittamalla jonokäsittelyn (tässä tyhjänä, IPCC hoitaa jononkäsittelyn) ja puhelun tarjoamisen agenteille. Koska tilaserveri kuvittelee "jononumeron" olevan kriittinen resurssi, se pyritään vapauttamaan nopeasti, ja tilaserveri lähettääkin IPX:lle **X_DIVERT_CALL**-viestin, jotta IPX siirtäisi puhelun hälyttämästä johonkin toiseen paikkaan. Tämä olisi PSTN-tapauksessa jononumero tai VRU-linja, mutta nyt **X_DIVERT_CALL**ia pidetään vahvistuksena sille, että CustomerConnect on alkanut käsittelemänn palvelupyyntöä. Ensimmäinen **X_DIVERT_CALL** välitetään edelleen MCU:lle **IS_DIVERT_CALL**in muodossa, ja vasta kun MCU on vastannut ilmoittaneensa IPCC:lle **IS_DIVERT_CALL_ACK**in myötä, vahvistetaan tämä tilaserverille **X_DIVERT_CALLCONF**-viestillä. Reitityksistä johtuvat useammat kutsunsiirrot IPX kuittaa vakiolla **X_DIVERT_CALL_CONF**illa, eikä lähetä MCU:lle mitään tietoa tästä.

Jossain kohdassa oikea agentti on löytynyt ja tämä on ottanut puhelun vastaan. Nyt PSTN-tapauksessa alaliittymää komennettaisiin vastaamaan puheluun, ts. tekemään kutsunsiirto jonotusnumerosta agentin alaliittymään ja siirtämään B-yhteys tilasta ALERTING tilaan CONNECTED: tilaserveri lähettää vaihdeserverille **X_ANSWER_CALL**in, ja nyt IPX tietää reagoida tähän siten, että on aika muuttaa virtuaalinen puhelu reaaliseksi. IPX lähettää MCU:lle viestin **IS_DIVERT_CALL**, ja tämä vuorostaan komentaa IPCC:tä luomaan yhteyden. Agentin päätteellä olevan IP-ohjelman on oltava *autoanswer*-tilassa vastaamisen onnistumiseksi [MICR97b] (mediavirta luodaan asiakkaan puheluohjelmalta päin). Agentin GUI:ssa vastaaminen – napin painaminen windows-sovelluksessa – avaa NetMeeting-ikkunan päällimmäiseksi yhdistetyssä tilassa.

Vastaamista ei katsota onnistuneeksi, ennenkuin MCU on palauttanut **IS_DIVERT_CALL-ACK**in positiivisin parametrein, jolloin IPX vuorostaan vastaa tilaserverin viestijonolle **X_ANSWER_CALL_CONF**illa – vastaus siksi, että tilaserverin viestit vaihdeserverille vaativat useimmiten synkronisen vastauksen.

Puhelun katkaisu tulee IPX:n kannalta kahdesta suunnasta: joko MCU:lta, nk. manuaalisena katkaisuna tai sitten tilaserveriltä USQ-järjestelmän pakottamana. Ensimmäinen kategoria käsittää IP-puheluohjelmista tehdyt puhelunkatkaisutoiminnot ja IPCC:n käyttöliittymästä lopetetut puhelut, kun taas jälkimmäinen tarkoittaa lähinnä agentin omasta CustomerConnect-clientin käyttöliittymästään katkaisemaa puhelua, joskin voi tulla myös muista, enimmäkseen poikkeustilanteisiin liittyvistä lähteistä. Periaatteessa IPCC:n käyttöliittymästä tehdyt katkaisuyritykset olisi mahdollista käyttää tilaserverin kautta, mutta tyypillisesti katkaisun epäonnistuminen yhdestä käyttöliittymästä johtaa asiakkaan vain käyttämään toista liittymää asian tekemiseen, jolloin kannattaa jo valmiiksi lähettää "manuaalinen katkaisu"- viesti tilaserverille.

Mikäli kysymyksessä on manuaalinen katkaisu, MCU lähettää IPX:lle **IS_CALL_HUNGUP**-viestin, joka muutetaan muotoon **X_MANUALLY_HUNGUP** ja annetaan edelleen serverille. Vahvistuksia näihin ei odoteta, koska viestit eivät pyydä lupaa vaan ilmoittavat tapahtuneesta.

CT-välitteinen katkaisu on pyynnön kaltainen: tilaserveri lähettää **X_HANGUP_CALL**in IPX:lle, joka kysyy MCUlta **IS_HANGUP_CALL**in avulla, voiko kyseisen puhelun katkaista. Jos MCU toteaa rakenteiden olevan kunnossa katkaisua varten, se lähettää **IS_HANGUP_GALL_ACK**in toiminnon vahvistavan parametrin kera, muutoin kyseinen viesti tulee toiminnon kieltävän parametrin mukana. Kun IPX on saanut jonkin tiedon pyynnön statuksesta, se antaa pyynnön edelleen eteenpäin tilaserverille **X_HANGUP_CALL_CONF**in muodossa.

5.4 Jatkokehitys

5.4.1 Ensisijaiset muutokset

Tietoturva on asia, jota on ajateltu tässä dokumentissa toistaiseksi vasta käsitteellisellä tasolla. Käytännössä logiikkakomponentit tulevat useimmiten sijaitsemaan eri puolilla palomuuria kuin MCU ja IPCC:t. Tämä asettaa joitakin rajoituksia palomuurille (yksityiskohtainen selvitys liitteessä A), kuten UDP- ja TCP-liikenteen läpipääsyn, sekä pakettikohtaisen filtteroinnin. Jonkinasteisen uhkan muodostavat ulkoapäin aloitetut konferenssit, jossa konferenssi-isännyys on asiakkaalla (vaikkakin kontrolli on CustomerConnect-serverillä). Jollei palomuuri pysty tätä käsittelemään, yhteys on joko mahdoton, tai saattaa paljastaa palomuurin sisäiset IP-osoitteet ulkopuolelle. Ideaalitilanne olisi konferenssin

initialisointi sisältäpäin ulos, jolloin isännyys ja konferenssin kontrolli ovat palomuurin sisäpuolella, ja toiminnallisuus ei olisi niin palomuurin tyypistä riippuvainen asia.

IP-systeemin herkin lenkki, eli palomuurin ulkopuolella toimiva keskitetty vaihde-komponentti - MCU on suojattu tietyiltä hyökkäyksiltä: MCU suorittaa autentikoinnin ennen liittymistään PBX:ään estääkseen vale-MCU:den kytkeytymisen. IPCC:kin omaa rajapinnat autentikointiin, mutta sitä ei käytetä kuin tietyille erikoiskohtelua saaville asiakkaiden joukoille. Tämä autentikointisysteemi on yksinkertainen tunnus-salasanapari, jota ei kryptata. Liikenteen kuuntelua käyttävän hyökkäyksen estämiseksi tunnus-salasanapari olisi sekoitettava jollakin epäsymmetrisellä salausmetodilla (kuten RSA).

Itse mediavirtoihin ei toistaiseksi pääse käsiksi ohjelmallisesti, mutta niiden kuuntelu on kyllä mahdollista, koska H.323 ei toistaiseksi käsittele salausta. TAPI-3.0:n mukana luvatut kryptoAPIt tarjoaisivat tähänkin mahdollisuuden [MICR97c], ja luultavasti myös NetMeeting kehittyy mediavirtojen kryptauksen tarjoavaan suuntaan [MICR97b].

Itse komponenteista voi sanoa sen verran, että IP-puolen API:sta johtuen viestiketjuja ei ole toteutettu täysimittaisina, ts. pyynnöt tarvitsisivat useimmiten kohteelta asti saadun vastauksen, mikä pätee tällä haavaa ainoastaan puhelun katkaisuun. Muissa tapauksissa vahvistusta ei jäädä odottamaan ketjun viimeiseltä lenkiltä (IPCC:ltä), vaan oletetaan operaation onnistuneen aina jossakin kohdassa ennen ketjun päätä. Tämä trans-aktionomaisuus on piirre, joka tulee lisätä luotettavuuden kasvattamiseksi. Lisäksi IPCC:n kohdalla säikeistys ei ole paras mahdollinen, mutta käytöliittymäkomponentille riittävä. Moniajo-ominaisuuksia ja synkronointia voi vielä kehittää tämänhetkisestäkin versiosta, kuten myös oliopohjaisuutta.

Arkkitehtuurissa H.323:n ja CSTA:n erilaisuus aiheuttavat sen, että IPCC:n ja NetMeeting-kontrollin kommunikointikanava voi vaatia lisäharkintaa. Tässä versiossa IPCC ja NetMeeting kommunikoivat samalla HTML-sivulla siten että IPCC ja NetMeeting muodostavat yhdessä terminaalin. CSTA-mallin kannalta tämä ei päde, koska NetMeeting-kontrolli edustaa enemmän puhelinlaitetta ja IPCC taas korkeamman abstraktiotason käyttöliittymää (ellei sitä sitten rinnasteta puhelimen "käyttöliittymään", mitä vastaan puhuu kuitenkin NetMeetingin oma GUI). Tällöin NetMeeting-kontrollin pitäisi olla suoraan vaihdekomponentissa kiinni, ja IPCC komentaisi sitä USQ-ydinkomponentin lävitse, ikäänkuin CustomerConnect-client tilaserverin kautta puhelinlaitetta. Tämä ero katsannossa vaatii enemmän pohdintaa, joskin NetMeeting-kontrollin ja IPCC:n eriyttäminen niinkin kauas toisistaan sitoo järjestelmän entistä tiukemmin CSTA-maailmaan, kun taas pyrkimys olisi siirtyä koko ajan enemmän H.323-yhteensopivaksi, milloin tulisi NetMeeting-kontrollin ja IPCC:n erottamisen sijaan muokata tila- ja vaihdeserverin rakennetta uudelleen.

5.4.2 Komponenttiarkkitehtuuri

Vaikka USQ:n kuvauksessa eri osaset on esitetty komponentteina, ne eivät sitä vielä tässä toteutuksessa ole: komponentti on määritelmän mukaisesti itsenäinen, paikkariippumaton logiikkayksikkö, jonka rajapinnat ovat avoimet ja vakiot minkä tahansa tätä rajapintaa tarvitsevan komponentin käyttää edellyttämättä mitään erityisiä kirjastoja tähän rajapintaan liittymiseen [NEVE95]. (Termiä on kyllä käytetty paikka paikoin ohjelmistokompo-nentin merkityksessä) CustomerConnect on tällä hetkellä rajapinnoiltaan avoin ja suhteellisen staattinenkin, mutta pystyäkseen liittymään tähän rajapintaan ohjelman on sisällytettävä itseensä jonkinasteinen määrä tietoliikennekerroksen infraa, mikä ei ole varsinaisen komponenttiarkkitehtuurin ominaisuus.

Komponenttimalliksi on CC:n tapauksessa valittava MS:n COM/DCOM, (Component Object Model / Distributed Component Object Model) koska se tukee parhaiten nykyistä kehitysympäristöä. DCOM-ajattelussa jokainen komponentti sisältää tietyt vakiorajapinnat ja *kaikki* kommunikointi komponentin kanssa tapahtuu tämän rajapinnan kautta. Itse komponentit voivat olla melkein mitä tahansa, kunhan ne toteuttavat julkistamansa rajapinnan. Paikkariippumattomuudesta ja liikennöinnistä alhaisella tasolla huolehtii nk. komponenttiväylä. DCOM tarjoaa tähän oman, OLE-pohjaisen (Object Linking and Embedding, COM:n esiaste) versionsa

Ensimmäisenä tehtävänä on tyypillisesti komponenttien identifioiminen. Koko USQ-järjestelmässä on erotettavissa viisi eri kategoriaa:

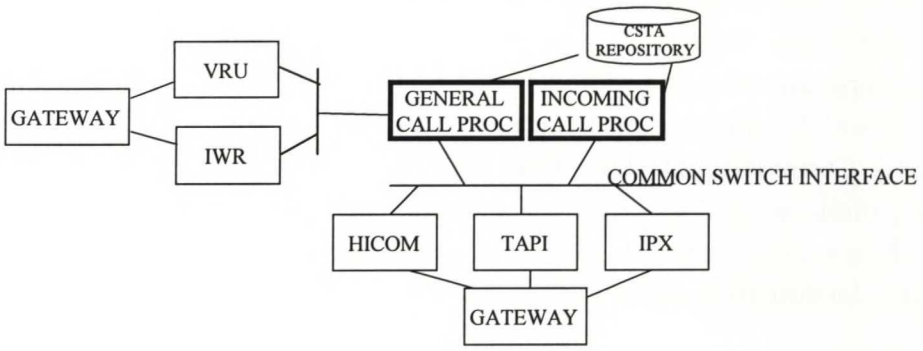
1. GUI-komponentit (käyttöliittymäkomponentit)
2. Palvelukomponentit (korkeamman tason palvelupyynnönohjausta tarjoavat komponentit)
3. Puhelunohjauskomponentit (keskitason USQ-toiminnallisuutta tarjoavat komponentit, kuten CSTA-malli ja IVR/IWR-systeemit)
4. Vaihdekomponentit (Kuten USQ-määrittelyssä)
5. Yleiseen tarkoitukseen käytetyt komponentit (DB)

Näistä kategorioiden 2 ja 3 huomataan vastaavan USQ-määrittelyn logiikkakomponentteja, erona ainoastaan IUR-komponentit. Kategoria 4 käsittää myös gateway-komponentit, mutta niiden ero vaihdekomponentteihin saattaa todennäköisesti muuttua veteen piirretyksi viivaksi lähitulevaisuudessa [PULV98].

Komponenttien keskinäisestä suhteesta tämän dokumentin puitteissa esitetään vain kategorioiden 3 ja 4 suhteet. Näiden välisistä rajapinnoista on esitetty jonkin verran USQ-määrittelyssä. Kategorioiden 3 ja 4 komponentit on esitetty kuvassa 5.6.

Rakenne on hyvin samankaltainen USQ-arkkitehtuurin kanssa, mutta toisaalta pyrkimyskin on juuri siihen suuntaan. CustomerConnect käsittelee toistaiseksi kahta mediaa, ja PSTN-

puolella kahta eri rajapintaa tottelevaa PBX:ää. Kaikille näille tulee pätemään sama vaihderajapinta, jota USQ:n logiikkakomponentit komentavat näkemättä mediatyyppiä.



Kuva 5.6: CC:n USQ-komponenttirakenne.

Tarvittaessa vihdekomponentit reitittävät puhelun yhdyskäytävä-komponentin kautta, jos A- ja B-laite ovat erityyppisiä medioiltaan.

Tulevien puheluiden käsittely on osoittautunut sen verran monimutkaiseksi, että se on järkevintä erottaa omaksi itsenäiseksi yksikökseen. Loput yksinkertaiset puhelutoiminnot, kuten ulospäin soitettava puhelu, yms. sijoitetaan samaan komponenttiin, GeneralCallProc-yksikköön. Näiden kahden logiikkakomponentin välinen yhteys tulee olemaan hyvin intiimi, joten on suotavaa, että ne sijaitsevat samassa prosessissa toistensa kanssa. Tilatieto talletetaan yhteen paikkaan nykyisen kopioinnin sijasta. Tämä estää ristiriitaisuuksien synnyn vaihdekomponenttien ja logiikkakomponenttien välille. CSTA-malli on kuitenkin edelleenkin paras malli laajamittaisessa puhelunohjauksessa, joten tilatieto säilytetään CSTA-mallin mukaisena erillisessä komponentissa, jonka synkronointi on erityisen hyvin ohjelmoitu sen sisäisen eheyden ylläpitämiseksi [HM&V97d].

IUR-komponentit ovat nekin logiikkakomponenttien käsiteltävissä, mutta tällä kertaa yhteisen rajapinnan kautta. Tarvittaessa IUR-komponentit tekevät siirtotiemuunnoksen jonkin yhdyskäytävä-komponentin avulla. On huomattava, että IUR-komponentit ovat usein vaihdekomponenttien yhteydessä, ja vaikka loogisessa mielesä käsketäänkin erikseen IUR-komponenttia, pyyntö saattaa loppujen lopuksi ohjautua vaihdekomponentille tai aiheuttaa ainakin mediavirran kulkemisen vaihdekomponentin lävitse.

5.4.3 Hajautettavuus

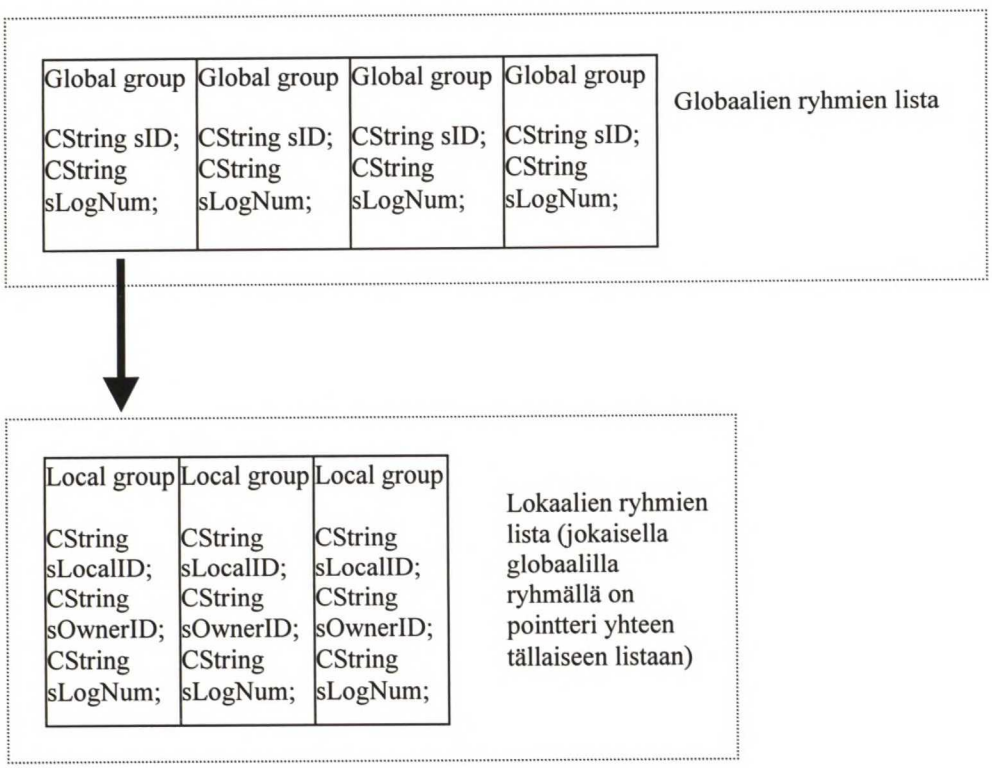
Yksi ohjausjärjestelmän tavoitteista oli hyvä hajautettavuus. Komponenttipohjainen arkkitehtuuri on jo iso askel tähän suuntaan, mutta käytännössä osoitehallinta ja reaaliaikainen ympäristö asettavat vaatimuksia, joita ei ratkaista pelkällä komponenttiajattelulla. Mukaan tulevat useat CustomerConnectin tilaserverit ja vaihdekomponentit, kuten esitettiin

USQ-osiossa. Periaatteena on, että resursseja voidaan jakaa, kuormaa tasapainottaa ja skaalata nopeasti. Tässä esitetty malli [HM&V97e] pohjaa H.323-tyyppiseen domain-ajatteluun, missä yksi domain koostuu yhdestä gatekeeperistää (CustomerConnectin tilaserveristä) ja siihen mahdollisesti liittyvistä muista USQ-järjestelmän osista. Tässä mielenkiintoista ei ole arkkitehtuuri, vaan USQ-palveluiden ryhmittely ja niiden käyttö.

Seuraavassa kuvataan kaksi lähtökohtaa palveluiden järjestämiselle hajautetussa ympäristössä: hierarkkinen ja rinnakkainen malli. Kummassakin käytetään domain-riippumattomia globaaleja ryhmiä ja alueisiin sidottuja lokaaleja ryhmiä. Ensinmainitussa mallissa globaalit ryhmät koostuvat lokaaleista, kun taas jälkimmäisessä globaalit ryhmät ovat lokaalien ryhmien yleistyksiä.

5.4.3.1 Hierarkkinen malli:

Hierarkkisessa mallissa on CustomerConnect 2.0:n ryhmiä vastaavia lokaaleja ryhmiä ja globaaleja ryhmiä. Jokainen globaali ryhmä koostuu joidenkin domainien tietyistä lokaaleista ryhmistä. Jokainen domain hallinnoi omaa ryhmäänsä, ja lisää/poistaa agentteja vain näihin ryhmiin/näistä ryhmistä. Jokaiseen ryhmään assosioidaan globaalisti yksikäsitteinen ID, ja globaalissa ryhmälistauksessa jokaisella globaalilla ryhmällä on lista niistä ID:stä, jotka sille kuuluvat [HM&V97e].



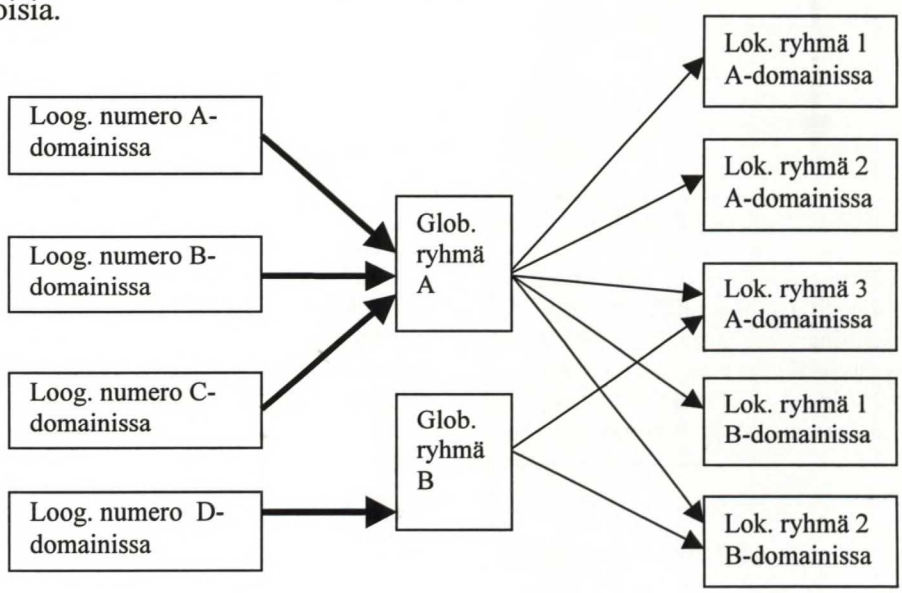
Kuva 5.7: hierarkkinen malli

Jokaisella lokaalilla ryhmällä on oma looginen numeronsa (PSTN & IP), ja jos lokaali ryhmä on globaalin ryhmän jäsen, tämä looginen numero on käytettävissä jokaisesta domainista. Lokaali ryhmä voi olla myös useamman globaalin ryhmän jäsen. Toisin kuten lokaaleilla ryhmillä, globaaleilla ryhmillä voi olla useita eri loogisia numeroita, kuitenkin vain yksi per domain.

Lokaalien ryhmien omistaja on se domain, jossa ryhmä on luotu ja jossa sitä pääasiallisesti on tarkoitus käyttää. Merkittävä tekijä erityisesti PSTN-maailmassa on myös se vaihdekomponentti, jonka alueella ryhmän looginen osoite (numero) sijaitsee. PBX on tässä tapauksessa tiukempi osoitteen omistuksesta kuin MCU. Jos ryhmä poistetaan tai lisätään, on operaatiosta ensin ilmoitettava muille domaneille ja vasta kun ryhmään operoivilta domaineilta on saatu kuittaus, voidaan ryhmämuutos tehdä. Jos jokin gatekeeper ei ole pystyssä sillä hetkellä, se ei lähetä kuittausta, eikä sitä odotetakaan kuin tietyn ajan. Edellämainitusta syystä jokaisen gatekeeperin on käynnistyessään ilmoittauduttava sellaisille domainille, joista se haluaa ryhmätietoja. Tässä tapauksessa, kuten luonnollisesti muissakin ryhmästatuksen muutoksissa MCU:ta on pidettävä CustomerConnect-clienttina siinä mielessä, että sen on oltava koko ajan selvillä nykyisestä ryhmätilanteesta.

Puhelunohjaus, PSTN

Puhelunohjaus paitsi PSTN-puolella, myös IP-puolella nojaa siihen, että jokainen domain on mahdollisimman suvereeni oman puhelumallinsa kanssa. Täten domainista uloslähtevät puhelut ovat yhteneviä yleiseen verkkoon suunnattujen puheluiden kanssa, vaikka niiden päätepisteenä olisikin mahdollisen kontrollin alla oleva alaliittymä. Vastaavasti yleisestä verkosta ja toisesta domainista saapuvat puhelut ovat vastaanottavalle domainille aivan samanarvoisia.



Kuva 5.8: loogisten numeroiden ja ryhmien väliset yhteydet

Globaalilla ryhmällä voi olla useampikin looginen numero, yksi domainia kohden. Kun tuleva puhelu hälyttää vaihteessa, loogisen numeron omistava CustomerConnect-serveri päättelee, mihin lokaaliin ryhmään puhelu pitäisi ohjata. Tämä tehdään kuorman tasapainotustekniikalla, jolloin jokainen globaaliin ryhmään osallistuva CustomerConnect-serveri pitää jonkinnäköistä statistiikkaa omasta kuormastaa, joka lähetetään säännöllisin väliajoin muille globaaliin ryhmään osallistuville domaineille. (Spontaani lähetys estää suorituskyvyn hävikkiä kovan kuorman aikana).

Jokainen domain vastaa myös omasta jononkäsittelystään (mukaanlukien IVR) sen jälkeen kun puhelu on reititetty globaalista ryhmästä lokaaliin. Kun puhelu on päässyt lokaalin ryhmän jonoon, se pysyy siinä, kunnes siihen vastataan tai se katkaistaan. Muut domainit eivät enää puutu tähän [HM&V97e].

Puhelut reititetään yksinkertaisella kutsunsiirrolla lokaalin ryhmän loogiseen numeroon. Serverien välistä liikennettä ei tässä tarvita, koska kutsunsiirron kohdedomainin serveri näkee tämän siirron vain uutena tulevana puheluna. Näihin lokaaleihin ryhmiin voi luonnollisesti myös soittaa suoraan, mutta riippuu numeron julkistamisesta, tapahtuuko näin koskaan. Tällaisen menetelmän varjopuolena on yhtäaikaisten puheluiden suurempi esiintymistiheys: jos kaksi domainia päättää siirtää puhelunsa samaan aikaan jollekin kolmannelle domainille, CustomerConnect:n tilaserverillä ei ole aikaa divertoida ensimmäistä jononkäsittelynumeroon, jolloin toinen kutsunsiirto epäonnistuu ja kontrolli tähän puheluun menetetään (puhelunkäsittelyssä jokainen domain oli itsenäinen). Tämän estämiseksi jokaisella ryhmällä (niin globaaleilla kuin lokaaleillakin) tulisi olla vaihteen sisäinen "varajärjestelmä", rivi analogisia muistipaikkoja PBX:n sisällä, jotka tunnetaan nimellä *hunt group*. Näihin muistipaikkoihin puhelu voidaan divertoida paljon nopeammin, koska tällöin vastuullisena osapuolena on itse PBX, joka laitteistototeutuksensa ansiosta on vastaavasti nopeampi kuin ohjelmistopohjainen ohjausjärjestelmä [HM&V97e].

Hajautetun järjestelmän vikasietoisuus tulee esille mm. seuraavassa järjestelyssä: PBX:ään on mahdollista määritellä nk. ennakkosiirtoja, jotka siirtävät puhelun hälyttämästä ennakkosiirtoon konfiguroidusta numerosta jonnekin toisaalle tietyn aikavakion kuluttua. Jos globaaliin ryhmään kuuluvien domainien loogiset numerot järjestetään tällaiseen ennakkosiirtojen ketjuun, saadaan tulokseksi looginen numero, joka voi ohjata palvelupyynnöt johonkin pystyssä olevaan domainiin, vaikkei itse tai seuraavakaan domain olisi toimintakunnossa. (Jos N:stä domainista jokaisesta yksi vapaita agentteja sisältävä ryhmä on liittyneenä globaaliin ryhmään, N-1 vapaasti valittua domainia voi olla käyttökelvottomia, ja järjestelmä pystyy silti käsittelemään puheluita, joskin muissa domaineissa paraikaa käynnissä olevat ja niihin juuri siirretyt puhelut menetetään domainin kaatumisen mukana.)

Vaikka globaalilla ryhmällä voikin olla useampi looginen numero käytettävissään, ryhmää voidaan myös käyttää siten kuin sillä olisi vain yksi looginen numero. Tässä voidaan joko staattisesti asettaa jokin domaineista primääriseksi ja jokin toinen sekundääriseksi, tai

käyttää jotakin yleisen verkon ohjauskomponenttia, kuten älyverkkoa, hajauttamaan ja tasaamaan kuorma.

Puhelunohjaus, IP

IP-puhelut eroavat PSTN-puheluista lähinnä siirtotiensä ja osittaisen virtuaalisuutensa puolesta. (Reaalisia vasta vastauksen jälkeen.) Tässä mitään keskitettyjä kuormantasaajia tai ohjelmistoa tukevia laitteistoja, kuten PSTN-puolen älyverkkoja ja PBX:iä ei ole olemassa. Tämä taas tekee kysymyksen IPCC:tä vastaavan clientin sijaintipaikasta ja jakelukanavasta ajankohtaiseksi. Koska clientin on tiedettävä muista domaineista jotakin, on nämä tiedot haettava käynnistysvaiheessa clientin kotidomainin tietokannasta. Clientin kotidomainina pidetään tässä sitä domainia, jonka alueelta (web-serveriltä) se on ladattu/haettu.

Jossain mielessä web-serverillä on samanlaisia ominaisuuksia kuin IN:llä PSTN-maailmassa: se on globaali osoitteiden jakelupiste, joka omalla tavallaan tasaa ja hajauttaa kuormaa, ja jonka luotettavuutta ei tässä aseteta kyseenalaiseksi.

Ryhmäpuhelun toteuttava client voi olla haettavissa jostakin globaalista web-serveristä, tai vastaavasti jostakin domain-spesifisestä serveristä. Ensimmäinen tapaus pätee globaaleilla ryhmillä operoiviin clientteihin, ja jälkimmäinen lokaaleja ryhmiä käsitteleviin. Ensinmainitut ottavat yhteyden johonkin primääriseksi määritellyn domainin MCU:hun, joka pystyy tuottamaan listan ja osoitetiedot kaikista globaaleista ryhmistä, ja viimemainitut omistajadomaininsa MCU:hun.

Kun client on ladattu, sen tekee kyselyn omaan domainiinsa globaalien ryhmien ryhmätiedoista, ja näyttää ne asiakkaalle latausparametrien mukaisesti suodatettuina. Kun asiakas on valinnut ryhmän ja aloittanut puhelun, clientin tulee määrittää, mille domainille se puhelun siirtää. Tässä voidaan soveltaa useita taktiikoita, jotka ovat parametroitavissa: client voi tehdä kyselyn kaikille domaineille (Kuva 5.9 a), joillekin niistä (Kuva 5.9 b), tai pollata jokaista erikseen vakiojärjestyksessä tai sattumanvaraisesti (Kuva 5.9 c). Jos kysely lähetetään kaikille yhtäaikaa, clientin tulisi omata jokin valintakriteeri yhtäaikaa saapuville vastauksille. Jos taas kysely tehdään jonossa, ei kannata odottaa kovin pitkiä aikoja.

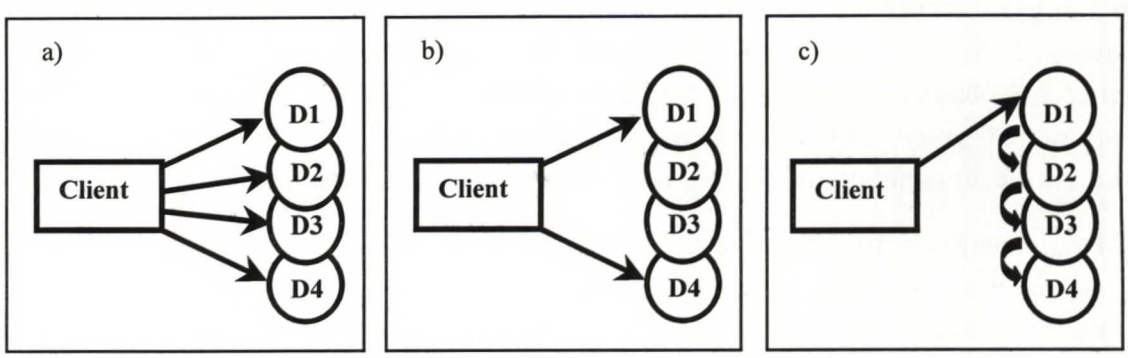


Fig. 5.9: Client-kyselyt useille domaineille

Domaineissa MCU:t ovat edelleen niitä komponentteja, jotka ottavat vastaan clientin kyselyt, ja päivittävät tiedot ryhmämuutoksista edelleen clienteille. Asiakkaan aloitettua puhelun yksikään domain ei omista sitä heti. Vasta kun client on valinnut vastaanottavan domainin, siirtyy puhelu jonkun CustomerConnect-serverin omistukseen. Tästä pisteestä eteenpäin puhelunkäsittely sujuu kuten PSTN-puolellakin sillä erolla, että PBX:ien tilalla on MCU:ita ja puhelu muuttu reaaliseksi vasta kun vastaanottavan agentin IP-osoite on löydetty.

5.4.3.2 Rinnakkainen malli

Tässä mallissa on myös lokaaleja ja globaaleja ryhmiä, mutta niiden välillä ei ole minkäänlaista viestintää. Globaali ryhmä on tässä mallissa aivan kuten lokaali ryhmä, paitsi että sillä on jokaisessa siihen osallistuvassa domainissa kaksi loogista PSTN-numeroa: sen lokaalia ryhmänumeroa vastaava numero (gln) ja muihin domaineihin suuntautuvaan reititykseen käytetty ylimääräinen numero (lln) [HM&V97e].

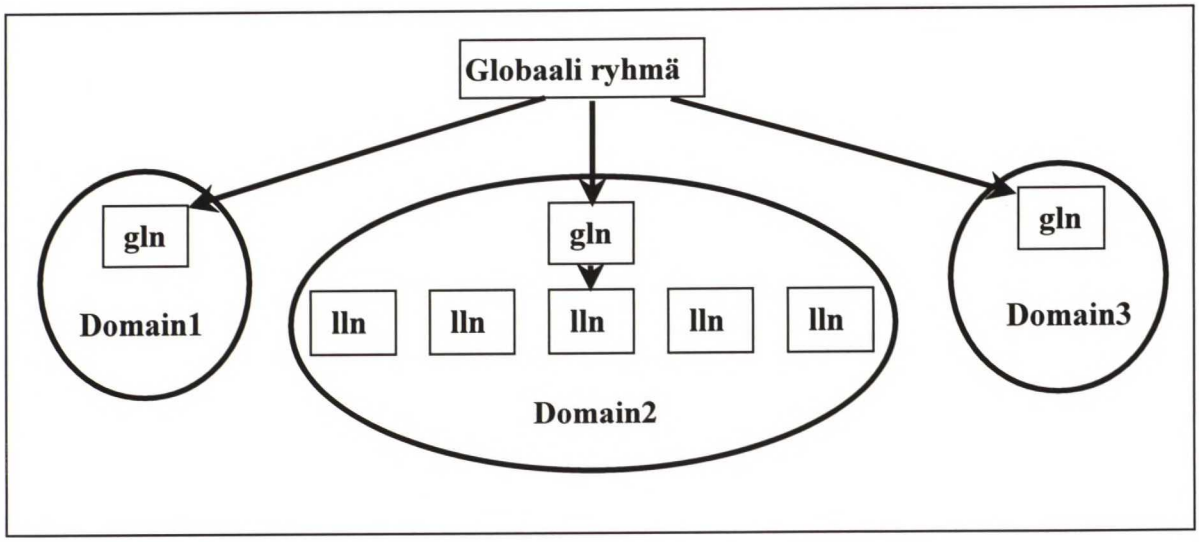


Fig. 5.10: rinnakkainen ryhmien malli

Tässä mallissa lokaalien ryhmien poistolla, lisäämisellä tai muuttamisella ei ole mitään vaikutusta muihin domaineihin, sillä ne eivät ole edes ulkopuolisten domainien välillisessä käytössä.

Rinnakkaisessa mallissa tasoja on vähemmän, mikä vähentää hallinnoinnin tarvetta, mutta poistaa toisaalta joitakin tärkeitä informaationlähteitä, jotka on korvattava vastaavilla kyselyillä ulkopuolisista domaineista. On esimerkiksi mahdotonta tietää, onko globaalissa ryhmässä yhtään agenttia vai ei kysymättä ensin kaikilta globaaliin ryhmään osallistuvilta domaineilta.

Puhelunkäsittely sujuu PSTN-puolella melko samaan tapaan rinnakkaisella kuin hierarkkisella mallillakin. Globaalilla ryhmällä voi taas olla useampi looginen numero (kuitenkin enintään yksi domainia kohden), ja CustomerConnect-serveri pystyy päättämään reitityksen kohteen jälleen muilta servereiltä saadun kuormitusdatan perusteella. Tässä mallissa puhelun kiertäminen globaalista ryhmänumerosta toiseen estetään reitittämällä puhelu toisten toisten domainien globaalia ryhmää vastaavaan lln:ään gln:n sijasta [HM&V97e].

Tämä mallikaan ei poista vanhaa toiminnallisuutta, vaan itse asiassa säilyttää ryhmä-agentti assosiaation: Agentit pystyvät kytkeytymään suoraan globaaleihin ryhmiin, mutta myös lokaaleihin ryhmänumeroihin voidaan soittaa suoraan.

IP-puheluiden tapauksessa suurin muutos on pikemminkin usemman domainin käsite kuin ryhmämalli. Itse asiassa asiakkaan clientille ryhmämalli ei näy mitenkään. Tämä johtuu siitä, että toisin kuten PSTN-puhelut, IP-puhelut eivät ole sidoksissa mihinkään fyysisiin muistipaikkoihin vaihteessa. Ne ovat jo siirtotiensä puolesta sopivia hajautettuihin ratkaisuihin. Toisaalta MCU:n täytyy jo ottaa huomioon ryhmämalli, koska se pitää yllä tietoja ryhmistä clienttia varten.

Rinnakkaisella ryhmämallilla on puolensa, kuten kerroksisuuden pienuus ja hallinnoinnin helppous, mutta myös varjopuolensa, joihin kuuluvat lisätyt muistipaikat vaihteissa – tyypillisesti arvokkaita resursseja - ja verkkoliikenteen kasvu johtuen hukkuneesta informaatiosta, jota on kyseltävä muilta domaineilta ahkerammin. Jos domainien määrä on suhteellisen alhainen, voidaan käyttää rinnakkaista mallia, koska silloin kustannukset sen eduista eivät vielä ole liian suuret. Domainien lukumäärän kasvaessa suuremmaksi niiden väliset yhteydet (ja tarvittava verkkoliikenne) kasvavat neliöllisesti, ja tarvittava ylimääräisten muistipaikkojen määrä lineaarisesti, joten vähänkään isommissa järjestelmissä hierarkkinen malli on kustannustehokkaampi.

6. Tulosten arviointi

Tämä työ on jaoteltu kolmeen osaan: standardi- ja rajapintaselvitys, itse USQ:n määrittely ja mahdolliset toteutusskenaariot, sekä erään USQ-toteutuksen läpivienti ja tarkastelu. Ensimmäinen osa on luonteeltaan kirjallisuustutkimusta, toinen osa määrittelyä ja kolmas osa ohjelmatyötä. Näiden arviointikriteerit ovat kullakin erilaisia. Kirjallisuustutkimuksessa tavoitteena on sopivien toteutustapojen ja mallien löytäminen. Määrittelyn tulisi puolestaan olla riittävän joustava salliakseen tarvittavia muutoksia ja toisaalta yksityiskohtainen ollakseen realistinen pohja sitä hyödyntäville toteutuksille. Itse toteutuksen arvioinnille on esitetty kriteerejä jo luvussa 5.1 (*Tavoitteet*), mutta päällimmäisenä tarkoituksena toteutuksella on kuitenkin olla toimiva ja laajennettavissa oleva, valittuja metodeja ja määrittelyjä hyväksikäyttävä palvelupyyntöjen käsittelijä.

Standardiselvityksessä on tutkittu kahta pelkästään suosituksiksi tai malleiksi tehtyjä standardeja (H.323 ja CSTA), kahta varsinaista rajapintaa (NetMeeting-API, TAPI 3.0), ja myös yhtä rajapinta- ja malliehdotuksen sisältävää suositusta (SCTP). Selvityksessä kahden ensimainitun kategorian suositukset on valittu lähempää tutkimusta varten yleisyyden ja soveltuvuuden perusteella. CSTA kuuluu itse asiassa USQ-toteutuksen "emäohjelmistona" käytetyn CustomerConnectin oletuksiin, mutta lähemmän tutkimisen jälkeen se osoittautui sopivaksi myös yleistetyn puhelun/palvelupyynnön ohjausjärjestelmän sisäiseksi malliksi. H.323 oli ensimmäisiä suurten ohjelmistotalojen tukemia alan järjestöjen tekemiä suosituksia. IP-puheluiden malleista H.323 gatekeepereineen näytti soveltuvan parhaiten IP-puolen malliksi. Kummastakin standardista on selvitetty sen hyviä ja huonoja puolia ja pohdittu, miksi kukin soveltuu oman mediansa alaiseksi. Tässä olisi ollut vielä mahdollista käsitellä jotakin muuta tarkoitukseen sopimatonta mallia osoittamaan näiden kahden edelläkuvatun mallin soveltuvuutta tehtäviinsä. Diplomityön puitteissa tämä olisi kuitenkin ollut tarpeetonta, koska jo kahden ensinkäsitellyn suosituksen vertailusta ilmeni joukko heikkoja kohtia, joita mallilla kussakin tarkoituksessa ei saa olla. (esimerkkinä 3rd-party-mallin monimutkaisuus vaihde- & päätetasolla ja toisaalta 1st-party-mallin riittämättömyys itse ohjauksen tasolla).

Rajapintojen osalta kuvaukset ovat tyypillisesti suppeampia kuin itse standardeissa/ suosituksissa. Tässä tapauksessa esitellyt API:t kuvastavat olemassaolevia rajapintoja (NetMeeting-API), USQ:lle sopivia tulevaisuuden rajapintoja (TAPI 3.0) sekä USQ:lle sopimattomia rajapintoja. Ei ole siis vain pitäydytty yhteen rajapintamahdollisuuteen, vaan on tutkittu muitakin mahdollisuuksia. Negatiivisena puolena voidaan mainita IP-puheluiden kehitysvauhti: tätä työtä aloitettaessa käytettävissä olevat rajapinnat ja itse USQ-komponentit olivat todella harvassa. Tämän kirjoitushetkellä H.323:n toteuttavia komponentteja alkaa olla jo kaupallisesti saatavilla (gatekeepereillä muutamia, yhdyskäytävillä kymmeniä ja erilaisilla terminaaleilla useita kymmeniä valmistajia). Olemassaolevia

rajapintojakin tulee epäilemättä olemaan näillä hetkillä muitakin kuin NetMeeting-API, mutta ne eivät välttämättä ole kovin yleisiä.

Universal Service Queuen määrittelyssä on ensin pureuduttu toiminnallisuuteen, ja sitten pohdittu erilaisia toteutusmahdollisuuksia USQ:lle. Määrittelyn lopullisen onnistumisen määrää se, kuinka erilaiset useita medioita käsittelevät järjestelmät USQ:hun mukautuvat, samoin kuinka yleispätevä itse käsite tulee olemaan. Käsite on sikäli tarvelähtöinen, että tyypillisesti erilaisissa viestinkäsittelyn solmuissa (yleensä CallCenter) on ollut kysyntää erilaisille kommunikointikeinoille puhelimella tapahtuvan viestinnän lisäksi. USQ:n määrittely on rakennettu tavallisen palvelupyyntöjonon logiikan pohjalta, ja pohdittu eri medioiden tuomia muutoksia tähän logiikkaan. Määrittelyä ei ole jätetty pelkän teorian tasolle, vaan se on viety arkkitehtuurin ja toimintakuvauksien tasolle, ja näiden kautta osoitettu, että logiikaltaan esitetty määrittely on realistinen ja toisaalta myös sisäisesti yhtenäinen.

USQ:n toteutusosio tukee sekin osaltaan määritelmää. Siinä on esitetty aiemmin kuvatun järjestelmän puitteissa erilaisia mahdollisia toteutuskeinoja kirjoitushetkellä uusimpien mahdollisten laitteistojen valossa (tässäkin IP-puhelumaailman kehitysvauhti alkaa muodostua takaiskuksi) ja osoitettu, että USQ-toiminnallisuus on mahdollista saada aikaiseksi varsin yksinkertaisin menetelmin. Tarkastelussa on nojauduttu useimpen sillä hetkellä olemassaolleiden H.323-komponenttien valmistajien mahdollisesti tarjoamiin tietoihin, sekä työn puitteissa suoritettuihin kokeiluihin mm. NetMeeting-APIn kanssa.

Määritelmä on suunniteltu hyvin modulaariseksi (komponenttipohjaisuus), joten uusien komponenttien lisääminen siihen ei tule olemaan ongelma. Viestipohjainen rajapinta sallii uusien viestien ja viestiryhmien lisäämisen ilman, että olemassaolevien toteutuksien toimintaan olisi tehtävä muutoksia. Määritelmä ei ota kantaa muihin toiminnallisuuksiin, kuin mitä USQ:lle on siinä määritelty. Erityisesti se siis ei sisällä CSTA:n SR-domainin ja H.323:n yhdyskäytävien takana olevien komponenttien kaltaista liittymää/määrittelyä, joten se on niiden suhteen joustava.

Määrittelyosiota on kytketty standardiosuuteen määrittelyosion lopussa esitellyllä standardivertailulla, jossa määrittelyn ja arkkitehtuurin eri osia on tarkasteltu eri standardien näkökulmasta.

Itse ohjelmatyössä on konstruoitu kaksi H.323-komponenttia, sekä yksi CSTA:n S-domainin komponentti. Tämän osan arvioitikriteereinä ovat ensisijassa toimivuus ja sen jälkeen laajennettavuus. Toimivuus on erikseen testattu, ja järjestelmä on havaittu toimivaksi, joskaan stressitestejä ei ole laitteiston vähyyden vuoksi juurikaan ollut mahdollista suorittaa. Tältä osin esimerkiksi MCU:n suorituskyky on vielä testaamatta. Sekä IP- että PSTN-puhelut näkyvät agentille samassa jonossa, ja tämä voi vastata niihin samalla tavalla näytöstä. Puhelun lopettaminen onnistuu myös yhtenevästi. Laajennettavuus useampaan mediaan kuuluu jo suunnitteluun: eri vaihdetunnisteet erittelevät eri medioita käsittelevät

vaihdekomponentit, itse tilaserveri ei ota kantaa mediaan ensinkään ja mediatyyppi on jäsenmuuttujana kaikissa CSTA-mallin objekteissa.

7. Yhteenveto

Tämän työn tavoitteena on ollut Universal Service Queue – käsitteen määrittely ja täsmentäminen sekä toteutus mukaanlukien toteuttamiseen ja kuvaamiseen käytettyjen mallien selvitys.

Luvussa 2 esitellään USQ:n alusta ja lähtökohdat. Tässä kuvataan tilannetta ja käsitteitä sellaisina kuin ne olivat ennen USQ:n käsitettä, ja joiden pohjalle itse ohjelmatyötä on lähdetty tekemään. Näistä kerrotaan Internetiin liittyvät USQ:n toteutuksen kannalta tärkeimmät käsitteet ja vastaavasti PSTN-maailmaan kuuluvat käsitteet.

Luvussa 3 käydään läpi USQ:n kuvaamiseen ja mallintamiseen tarvittavia standardeja kirjallisuustutkimuksen muodossa. Tarkoituksena on perehdyttää lukija myöhemmin tarvittavaan käsitteistöön ja toisaalta auttaa ymmärtämään eri toteutuksessa ilmenevien ratkaisujen syitä ja seurauksia. Rajapintojen selvityksessä on tarkoituksena edellisten lisäksi myös osoittaa suuntaviivoja jatkokehitykseen.

Luvun 4 määrittelyosiossa syvennetään itse USQ:n käsitettä ja pohditaan mahdollisuuksia sen toteuttamiseen, tarkoituksena kirkastaa USQ:n kuvaa ja osoittaa sitä tukevien järejestelmien konstruointikeinoja. Eri toteutusskenaarioissa on pyritty ottamaan huomioon todelliset järjestelmät ja toteutusmahdollisuudet näiden pohjata. Tämän luvun pohjalta lukijan pitäisi pystyä myös arvioimaan luvun 5 toteutusta. Aivan lopussa on verrattu USQ:n toiminnallista määrittelyä eri standardeihin.

Luku 5 on alkuosaltaan tehtyä ohjelmatyötä esittelevä johdanto, jonka tarkoituksena on antaa ohjelmistotekniikkaan perehtymättömällekin lukijalle jonkinlainen käsitys järjestelmästä. Suurin osa tästä luvusta on kuitenkin teknispainotteista kuvausta eri komponenteista ja niiden toteutuksesta. Siinä osoitetaan toteutuksen toimivuus ja toisaalta järkevyys. Luvun loppu on käytetty huomioihin jatkokehityksestä suurimpana osanaan luvussa 4 esiteltyä USQ:n pilvimallia tukeva kuvaus CustomerConnectin hajautettavuudesta.

Luvussa 6 on pyritty arvioimaan työn eri osia ja perusteltu joitakin tässä työssä käyttöönotettuja ratkaisuja.

8. Lähteet

Kirjalliset lähteet:

DAVI97 Davidson, P; McConnell, B; Lung, E; Gelbach, S; Pines, J. *"Simple Computer Telephony Protocol (SCTP) version 0.11"*, 1997.

DAWS96a Dawson, K. *"What is a Call Center?"* CallCenter, August 1996, No. 8, ss 34 – 39.

DAWS96b Dawson, K; Lenz, M. *"The Call Center Leadership Awards"*. CallCenter, August 1996, No. 8, ss 40 – 54.

DAWS96c Dawson, K; Lenz, M. *"How we got here & Where we're going"*. CallCenter, August 1996, No. 8, ss 64 – 87.

EFUS97 eFusion Inc. *"eStream™ Enhanced Internet Services Application Gateway"*. eFusion Inc, 1997.

ECMA94 European Computer Manufacturer's Association (ECMA). *"Services for Computer Supported Telecommunications Applications, 4th draft"*, ECMA, 1994.

HANN96 Hannus, J; Huomo, T; Korhonen, J; Kortesniemi, A; Lamminmäki, S; Mäkelin, M; Vuoria, A. *"Internet ja intranet yritystoiminnassa – Visio, soveltamiskohteet, tekniikat, menetelmät"*. HM&V Research Oy, 1996

HM&V97a HM&V Telecommunications. *"CustomerConnect asennusohje, Versio 1.5"*. HM&V Telecommunications Oy, 1997.

HM&V97b HM&V Telecommunications. *"CustomerConnect käyttöohje, Versio 1.5"*. HM&V Telecommunications Oy, 1997.

HM&V97c HM&V Telecommunications. *"CustomerConnect pääkäyttäjän ohje, Versio 1.5"*. HM&V Telecommunications Oy, 1997.

HM&V97d HM&V Telecommunications. *"CustomerConnect component architecture"*. HM&V Telecommunications Oy, 1997.

HM&V97e HM&V Telecommunications. *"CustomerConnect Distributability"*. HM&V Telecommunications Oy, 1997.

HM&V98 HM&V Telecommunications. *"CustomerConnect Technical Documents"*. HM&V Telecommunications Oy, 1998.

HUOM93 Huomo, T; Mäkelin, M. *Palveluprosessit ja monikanavamarkkinointi, Suorat asiakassuhteet uuden teletekniikan avulla.* HM&V Research Oy, 1993.

ITUT96 ITU-T. *"Visual Telephone Systems and Equipment for Local Area Networks Which Provide a Non-guaranteed Quality of Service, draft".* ITU-T, 1996.

MICR97a Microsoft. *"Microsoft ActiveX Control Reference for NetMeeting".* Microsoft Corporation, 1997.

MICR97b Microsoft. *"NetMeeting 2.0 Resource Kit".* Microsoft Corporation, 1997.

MICR97c Microsoft. *"Windows ® Operating System: IP Telephony with TAPI 3.0 White Paper".* Microsoft Corporation, 1997.

NEVE95 Nevermann, P; Oppel, K; Thomann, J; Whitehead, K; Yourdon, E. *"Mainstream Objects: An Analysis and Design Approach for Business".* Prentice Hall Inc., 1995.

SIEM95 Siemens. *"Hicom 300 CallBridge/ACL Applications Connectivity Link: Beschreibung"*, Siemens AG, 1995.

STAL94 Stallings, W. *"Data and Computer Communications".* MacMillan Publishing Company, 1994.

On-line – lähteet:

DIAL97 Dialogic. *"Dialogic System Software and SDK for Windows NT"* http://support.dialogic.com/releases/winnt/9708/Release_Catalog, Dec 2, 1997.

MICR98 Microsoft. *"How to Establish NetMeeting Connections Through a Firewall"* http://support.microsoft.com/support/kb/articles/q158/6/23.asp, Oct 22, 1998.

PULV98 Pulver.Com. *"pulver.com IP Telephony Resource"* http://www.pulver.com/gateway, Nov 1, 1998.

WEBB96 Webb, K. *"Developing ISAPI Extensions"*, http://www.iftech.com/classes/isapi, Jul 18, 1996.

SUN98 Sun Microsystems. *"What You Should Know About Sun v Microsoft"*, http://www.javasoft.com/lawsuit, Dec 2, 1998

Liitteet

Liite A: CustomerConnect 2.0 IP-yhteys: Tekninen dokumentti

Liite B: CustomerConnect 2.0 IP-yhteyden viestien vastaavuus H.323:n RAS ja puhelusignalointikanavien viestien kanssa

Liite A

CustomerConnect 2.0 IP-yhteys: Tekninen dokumentti

1 Sovelluskomponentit ja rajapinnat

1.1 IP Customer Client

1.1.1 Käyttöliittymä

IPCC käyttöliittymä on parametrisoitavissa monipuolisemman toiminnan vuoksi. Parametrit välitetään IPCC:n sisältävällä HTML-sivulla. Parametrit on listattu edellä, ja tärkeimmät niistä ovat kolme listaa. Ensimmäisen tunniste on "`Acc_groups`", ja siinä määritellään ne palveluryhmien ID:t, joihin tästä appletista on oikeus ottaa yhteys. Ryhmien ID:n tulee olla samoja kuin ryhmädataa ylläpitävässä tietokannassa. Kaksi muuta listaa eivät ole pakollisia: "`ID_fields`" ja "`IDOb_fields`". Nämä määrittelevät käyttäjältä vaaditut tunnistetiedot, ennen kuin hänen sallitaan ottaa yhtes CustomerConnect-järjestelmään.

Yllämainitut parametrilistat muokkaavat käyttöliittymän ulkonäköä siten, että jos tästä kyseisestä appletista saavutettavien ryhmien määrä ylittää ennalta-asetetun rajan (tällä hetkellä N=3) käyttäjälle näytetään ListBox-kontrolli, ja muussa tapauksessa 1..N ryhmien nimien identifiiomaa nappia. Jos ID-listat ovat määriteltyjä, käyttäjälle annetaan mahdollisuus antaa itsestään joitakin tunnistetietoja. Haluttu informaatio esitetään tällä hetkellä staattisina rekisteröintikenttien indeksinä, josta applet tulkitsee kullekin indeksille asetetun selitteen. "`ID_fields`"-lista sisältää kaikki mahdolliset indeksit, joita asiakkaalta voidaan pyytää. "`IDOb_fields`"-listan taas on oltava osajoukko viimeksimainitusta listasta, ja sen ilmoittamien indeksien avulla kerrotaan ,mitkä kentät asiakkaan on pakko täyttää, ennenkuin on mahdollista avata yhteys CustomerConnectiin.

Jokainen lista on merkkijono, jonka formaattina on pilkuilla erotettu lista joko ei-negatiivisia numeroita tai tällaisina ilmaistuja indeksien joukkoja. Indeksijoukko ilmaistaan seuraavasti: `[alkuindeksi (mukaanlukien)]-[loppuindeksi (mukaanlukien)]`. Esimerkkinä: `IDOb_fields = "1,4,7,9-14,17"`.

Rekisteröintikenttien Tunnukset mapataan kenttien nimiksi taulukossa A.1 esitetyllä tavalla

Taulukko A.1: Rek.kenttä ID -> Rek.kentän nimi

ID	Nimi
0	Asiakasnumero
1	Yrityksen nimi
2	Toimiala
3	Koko
4	Henkilo

5	Asema
6	Osasto
7	Puh
8	Email
9	Fax

Käytännössä tämä rajapinta voi olla mitä vain yksittäisestä napista (saadaan aikaiseksi määrittelemällä vain yksi ryhmä-ID, ja jättämällä muut listat määrittelemättä) dynaamiseen listaan rekisteröintikenttineen.

Debuggausta varten käyttöliittymässä voi olla myös kaksi erillistä nappia, ja Listbox-kontrolli debug-viestejä varten: jos HTML-tiedostossa on parametri nimeltä "`IncludeDebugGUI`", joka voidaan myös asettaa TRUEksi tai FALSEksi, käyttöliittymään ilmestyy muun lisäksi kaksi nappia ("soita" ja "katkaise"), sekä TextBox-kontrolli, joden avulla voi testata NetMeeting-APIa suoraan ilman CustomerConnectin väliintuloa. Listbox-kontrolli on eräänlainen debug-ikkuna selaimille, jotka eivät tue java-konsolia.

Parametrit ja niiden merkitys on selvitetty seikkaperäisemmin allaolevassa taulukossa A.2.

Taulukko A.2 IPCC:n parametrit

Parametrin nimi	Pakollinen	Arvoavaruus	Tarkoitus
`IPSoftw`	**Ei** (oletuksena 0)	Ei määritelty	(ei käytössä)
`Acc_groups`	**Kyllä** (tämän jättäminen tyhjäksi ei kaada ohjelmaa, mutta asiakkaalle ei ilmaannu valittavaksi yhtään ryhmää, jos tämä puuttuu.)	MS-Word tyyppinen lista kokonaislukuja, esim. "1-4,7,9,12-15". Väärä formaatti heittää poikkeuksen, ja applet ei suostu suorittamaan.	Ryhmätunnisteiden lista, jonka mukaan näytetään ryhmät UI:ssä.
`Get_groups`	**Ei** (oletuksena *false*)	*True* tai *false*. Jos tekstinä jokin muu, alustuu false:ksi.	Bitti, jonka perusteella päätetään, haetaanko ryhmät MCU:lta vai ei.
`StandAloneDebug`	**Ei** (oletuksena *false*)	*True* tai *false* Jos tekstinä jokin muu, alustuu false:ksi.	Bitti, jonka perusteella päätetään, kytkeydytäänkö MCU:hun. Jollei, generoidaan

			demo-ryhmät.
IncludeDebugGUI	**Ei** (oletuksena *false*)	*True* tai *false* Jos muu teksti, alustuu falske:ksi.	Bitti, jonka perusteella päätetään, käytetäänkö debug-UI:tä.
ID_fields	**Kyllä** (tämän poisjättäminen ei kaada applettia, mutta asiakkaalle ei ilmesty tunnistetietokenttiä)	Kuten Acc_Groups	Esitettävien tunnistekenttien indeksilista.
IDOb_fields	**Ei** (tämän poisjättäminen tarkoittaa, että jokainen tunnistekenttä on vapaaehtoinen täyttää.)	Kuten Acc_Groups	Pakollisten tunnistekenttien indeksilista.
MCU_Addr	**Kyllä** (tämän poisjättäminen ei kaada applettia, mutta MCUta ei löydetä)	Mikä tahansa IP-osoite neljän pisteillä erotetun kokonaisluvun [0..255] muodostamana merkkijonona.	MCU:n IP-osoite (osoite, jota sen viestijono kuuntelee)
MCU_Port	**Kyllä** (tämän poisjättäminen ei kaadak applettia, mutta MCUta ei löydetä)	Mikä tahansa kokonaisluku, yleensä standardiporteista ylöspäin jonkin matkaa [1024...65536]	MCU:n viestijonon kuuntelema portti.
Offline_mail	**Kyllä** (tämän jättäminen pois ei kaada ohjelmaa, mutta osoitetta ei alusteta mihinkään tiettyyn arvoon joten sähköposti ei lähde minnekään)	Mikä tahansa merkkijono (olemassaoleva sähköpostiosoite)	Sähköpostiosoite, jonne asiakas voi lähettää kysymyksen MCU:n ollessa alhaalla.
Demo_IWR	**Ei** (oletuksena *false*)	*True* tai *false* (Muut merkkijonot alustuvat *false:ksi*)	Bitti, jonka mukaan päätetään, näytetäänkö IWR-jononkäsittely kokonaan ennen puhelun aloittamista.

			Vain ilman MCUta tapahtuvaan demoon.

1.1.2 AXI – rajapinta

Ne ohjelmistot, jotka käsittelevät varsinaista mediavirtaa, ovat Microsoft NetMeeting 2.1 kommunikointiin tapahtuvaa mediavirtaa varten ja RealMedia 5.0 IWR-viestejä varten. AXI kommunikoi NetMeetingin ja RealMedian COM-objektien kanssa eräänlaisten java-kääreiden avulla (wrapper). Tässä tekniikassa kommunikointi tapahtuu kehitystyökalun generoimien `.class`-tiedostojen avulla. (MS DevStudio 5.0:ssa oleva Java TLB Wizard käy läpi kaikki ActiveX-objektien (haetaan .OCX, .OCA ja .DLL-tiedostoista) tyyppikirjastot, ja tuottaa näiden avulla javan metakoodia käärittävien luokkien ActiveX-objektien rajapintoja vastaavasti. AXI käyttää näitä generoituja java-luokkia kommunikoidakseen varsinaisten COM-objektien kanssa.

AXI:n luokkia ja funktioita kutsutaan kahdesta paikasta, joita ovat 1) käyttöliittymä ja 2) socket-yhteys. AXI:n tehtävänä on toimia viestinvälittäjänä ja tulkkina näiden kahden IPCC:n osan välillä. UI kutsuu COM-luokkien ympärillä olevien java-luokkien vastaavia metodeja, ja socket-yhteyden kautta AXIa kutsuu MCU samalla tavoin kuin UI. MCU käyttää socket-yhteyttä CustomerConnectin tietoliikennekirjastojen käyttämän formaatin kautta. Tämä on toteutettu erillisissä viestikerroksen funktioissa socket-yhteyden sisällä.

AXI saa RealMedia- ja NetMeeting-COM-luokkien ilmoittamat viestit käyttämällä VBScriptillä kirjoitettuja tapahtumankäsittelijöitä, jotka vuorostaan kutsuvat itse appletissa olevia funktioita. On huomattava, että elleivät java- ja COM-luokat sijaitse samalla fyysisellä HTML-sivulla, VBScript-tapahtumankäsittelijöille ei lähetetä viestejä halutuista NetMeeting- ja RealMedia-objektien tapahtumista. (Ongelma liittyy oikean COM-luokan instanssin löytämiseen). HTML-sivun latausvaiheessa VBScriptiä käytetään antamaan COM-luokkien osoitteet appletille, joka ei luo omia instanssejaan objekteista, vaan tallentaa näiden HTML-sivuilla sijaitsevien instanssien pelkät osoitteet. Kun jompikumpi COM-objekti lähettää viestin, se kulkeutuu johonkin VBScript-tapahtumankäsittelijöistä, jos sellainen on lähetetyn tyyppiselle viestille määritelty, ja sieltä edelleen appletille. Syy tapahtumankäsittelijöiden tekemiseen VBScriptillä on juuri tässä vaiheessa: COM:in ja Javan käyttämät tietotyypit eivät ole suoraan yhteneviä, vaan täytyy tehdä käännös VBScriptillä tehdyissä funktioissa. Edelleen ainoastaan sellaiset viestit, joita kyseisen COM-objektin API tukee on ylipätään mahdollista kaapata. (Tuetuista ja tukemattomista viesteistä ks. päätekstin NetMeeting-APIsta)

NetMeeting-COM – sovelluksen yhteydessä on kehitystyötä varten ensin hankittava siihen kuuluva SDK (Software Development Kit), jotta olisi mahdollista istuttaa objekti eri sovellusten yhteyteen. Tämä SDK sisältää C++ - API:n, jossa on kaksitoista funktiota C-kehittäjille, COM-objektin ja ActiveX-komponentin. (API ja COM-objekti on

päivitetty versiosta 1.0 versioon 2.0, mutta ActiveX-komponenttia ei – eikä ole ilmeisesti aikomuksiakaan päivittää. Tämä hidastaa kehitystyötä tuntuvasti, koska video- ja audiokonferensoinnissa on useita eri toimintoja, joihin ei päästä käsiksi version 1.0 ActiveX-komponentista, mutta kylläkin version 2.0 COM-objektista.) SDK tarvitsee ladata vain kehitystyötä, ei käyttöä varten. Appletin käyttäminen vaatii vain NetMeeting-COM-objektin, joka tulee ohjelmiston standardipaketin mukana. Asiakkaan päähän ei siis tarvita SDK:ta vaan ainoastaan NetMeeting.

NetMeetingin SDK:n tukema toiminnallisuus sisältää tällä hetkellä **MAKE_CALL**- ja **HANGUP_CALL**-komennot, sekä useita viestejä, joista vain **MEMBER_CHANGED**-viestiä käytetään puhelun katkaisun havaitsemiseen. Puhelunsiirto on tarkoitus sisällyttää mukaan heti, kun NetMeeting-API:n lävitse saadaan kulkemaan komento mediavirran siirtämisestä konferenssissa. Jo pelkästään kahdella tuetulla funktiolla ja oikealla ajoituksella voidaan toteuttaa monia ACD-toimintoja.

RealMedian ActiveX-kontrolli tulee ohjelmistopaketin mukana, joten tämän yhteydessä ei tarvitse ladata erikseen mitään SDK:ta. RealMedia-kontrolli käyttää kohtuullisen kokoista komentojoukkoa, joista vain muutamia käytetään tämän sovellutuksen yhteydessä. Käytetyt komennot ovat **DO_STOP** and **PLAY_PAUSE**, ja viestit (eventit) **CLIP_CLOSED**. Komentoa **PLAY_PAUSE** käytetään IWR-viestin käynnistämiseen ja **DO_STOP** samaisen viestin lopettamiseen, mikäli asiakas keskeyttää jonottamisen tai yhteys agenttiin saadaan syntymään. Molemmilla em. komennoilla on lisäksi **CAN_** -etuliitteillä varustetut funktiot, joiden avulla voidaan tarkistaa, pystyykö RealMedia-COM-objekti suorittamaan halutun komennon. **CLIP_CLOSED**-tapahtumankäsittelijää käytetään selvittämään IWR-viestin loppumista. Tätä hyödynnetään kahdessa tapauksessa: a) viestin aloittamiseen uudelleen, jos jonotus kestää pitkään ja b) käynnistämään mediavirta NetMeetingin avulla ilman IPX:ää tapahtuvassa demoympäristössä.

AXI suorittaa pääsäikeen yhteydessä, samassa säikeessä kuin UI. Ennen kuin applettia voi ajaa, on otettava huomioon useita seikkoja, jotka käsitellään tarkemmin luvussa 1.1.4 appletin asennuksesta.

1.1.3 Socket-yhteys MCU:hun

Socket-yhteys MCU:hun suorittaa eri säikeessä kuin käyttöliittymä ja AXI. Sen päätehtävänä on odottaa MCU:n yhteydenottoa ja tämän jälkeen kuunnella verkkoa MCU:n lähettämien viestien varalta ja lähettää saamansa mahdolliset komennot edelleen AXI:lle tai UI:lle. Tämän säikeen suorittava osuus on päättymätön silmukka, joka saa viestin, parsii sen ja kutsuu vastaavaa AXI:n funktiota. Tämän yhteyden kautta kulkevat viestit menevät suoraan CustomerConnectin viestipohjaisen tietoliikennenkirjaston rajapinnan kautta TCP/IP-socketin lävitse. Tämän vuoksi IPCC sisältää samanlaiset funktiot, luokat ja rakenteet alatason merkkijonon käsittelyyn kuin CustomerConnectin tietoliikennekirjasto. (Huom: vaikkakin infran puolesta IPCC pystyisi käyttämään myös CustomerConnectin A-rajapintaa, arkkitehtuuri ei sallisi A-rajapinnan koko ilmaisu-

voiman hyödyntämistä, joten IPCC:lle on määritelty oma rajapintansa: iC-rajapinta, joka on dokumentoitu MCU:n yhteydessä.

Socket-yhteys kuuntelee verkkoa siinä dynaamisesti asetetussa portissa, jonka IPCC:n käyttöjärjestelmä on sille määrittänyt. Kun kuunteleva säie huomaa viestin, se synnyttää uuden säikeen, joka ensin konstruoi saamastaan merkkijonosta sen tyypin ilmoittaman viestiluokan ja toimittaa sen eteenpäin kutsumalla asianmukaista AXI:n funktiota viestiluokan `process()`-metodista. Verrattuna CustomerConnectin vastaavaan toimintaan socket-yhteys vastaa client-puolelle tehtyä `ReceiveThread`iä. Viestiluokan koodaus ja dekoodaus merkkijonoksi tehdään luokan konstruktoreissa: vastaanottavissa viesteissä dekoodaus ja lähettävissä viesteissä koodaus. IPCC:ssä ei ole erillistä viestijonoa, koska se on toistaiseksi liian raskas tarkoitukseen.

Socket-yhteyden luokissa on myös metodit datan lähettämiseen allaolevan TCP/IP-socketin lävitse. Itse socket-yhteyden säie ei tätä vartioi, vaikkakin se kuuluu samaan instanssiin socket-säikeen kanssa. AXI kutsuu tätä metodia, kun sen tarvitsee lähettää asiakkaan UI:n kautta ilmoittamat toimenpiteet edelleen MCU:lle. Tällöin konstruoidaan asianmukaisen `JMsg`-luokan instanssi, ja kutsutaan sen `sendToServer()`-jäsenfunktiota. `SendToServer()` vuorostaan luo itse viestin ja sarjallistaa sen (muokkaa sen merkkijonoksi) ennen itse socket-yhteyden `send()`-metodin kutsumista.

Lähettäminen tehdään sikäli sokeasti, että parametrejä ei juuri tarkistella. Asiakas ei nimittäin pysty lähettämään mitään suoraan GUI:sta, ja toisaalta itse rajapintakin on pelkästään HM&V:n käyttöön tarkoitettu, joten rajapinnan väärinkäytön mahdollisuudet eivät ole suuret. Vastuu parametrien oikeellisuudesta on näinollen ohjelmoijalla.

1.1.4 Asennus

1.1.4.1 Applet

Ennenkuin appletin voi laittaa web-palvelimelle ladattavaksi, se on pakattava sovelluskehittäjän identifioivan digitaalisella allekirjoituksella varustettuun nk. cabinet-tiedostoon (`.CAB-file`) Tämän luomisesta enemmän kappaleessa "*Creating and signing files for distribution over the internet*" dokumentissa *CustomerConnect Technical Documentation.*

Kun applet on pakattu `.CAB`-tiedostoksi, siihen pääsee käsiksi HTML-sivulta normaalin `<applet>`-merkinnän avulla. Tähän pitää muiden parametrien lisäksi kirjoittaa uusi cabbase-parametri. Koko merkintä näyttää esimerkiksi allaolevan mukaiselta:

```
<applet code=CliCtrl.class>
<param name="cabbase" value="cabfilename.cab">
</applet>
```

Appletin ja digitaalisen allekirjoituksen sisältävä `.CAB`-tiedosto on sijoitettava `cab-base`-parametrin osoittamaan paikkaan. Kabinettitiedostot tunnistaa vain Microsoftin Internet Explorer 4.0 tai sitä myöhemmän version selain. Appletia käyttävällä asiakkaalla on oltava NetMeetingin vähintään version 2.0 ja RealMedian vähintään version 4.0 COM-objekti asennettuna ja rekisteröitynä koneellaan. Jollei tällaista ole, selain ilmoittaa tästä automaattisesti asiakkaalle, ja tarjoutuu lataamaan kyseisen komponentin HTML-sivulla ilmoitetusta osoitteesta.

Jos edellä mainitut vaatimukset on täytetty, palveluun kytkeytyvän asiakkaan tarvitsee vain ladata HTML-sivu joltakin enneltamääritellyltä palvelimelta. Kun applet on purettu kabinettitiedostosta, se ajetaan automaattisesti samalla tavalla kuin pakkaamatonkin applet.

1.1.4.2 IWR (RealMedia)

Koko IWR-järjestelmä koostuu viestistä (viestitiedostosta), sitä käsittelevästä lähettimenä toimivasta RealMedian serveristä, ja vastaanottimen roolissa olevasta RealMedia Playerin COM-komponentista, joka sijaitsee samalla HTML-sivulla IPCC:n kanssa. Normaalisti RM-viestitiedoston voi soittaa ilman RealMedia-serverinkin väliintuloa, mutta ActiveX-komponentin tapauksessa ainoa käsiteltävä protokolla on streaming-protokolla `pnm`, jonka tuottamiseen tarvitaan RealMedian streaming-palvelin.

RealMedia-serverin on oltava samalla palvelimella kuin web-palvelinkin käytetyn osoitteistuksen vuoksi (RM-serveri tunnistaa protokollan, mutta ei pidä yllä omaa osoitteistustaan). Käytettyjen streaming-tiedostojen tulee olla saatavissa web-palvelimen juurihakemiston kautta (jos esim. Web-palvelimen juuri on `C:\InetPub\wwwroot\`, RM-tiedoston tulee olla siellä tai jossakin sen alihakemistoissa). Tällöin RM-tiedostoihin viitataan pnm-protokollalla web-palvelimen osoitehierarkialla: `C:\InetPub\wwwroot\rmfiles\`:n alla olevaan tiedostoon `queue_msg.rm` viitataan osoitteella `pnm:\\www.company.com\rmfiles\queue_ms.rm`, jos web-serverin juuri on kuten aiemmin kuvattiin ja tarjoaa www-palvelua osoitteen www.company.com alla. Ennenkuin IWR on toimintakunnossa, RealMedia-serveri on startattava. Tämä tapahtuu Serveripaketin mukana tulevan Real Server Control Centerin kautta. Itse RealServerin ajotiedoston nimi on `server.cfg`.

RealMedia-COM-objekti on osa RealMedian alasladattavaa ohjelmistopakettia, ja kun tämä puretaan, COM-objekti installoituu ja rekisteröi itsensä automaattisesti. COM-objektin voi sijoittaa HTML-sivulle parhaiten ActiveX Control Pad:in avulla, jolloin luokka-ID:n selvittämisen voi jättää varusohjelman huoleksi. Itse objektilla on useita parametrejä, ja tarvittu HTML-merkintä on esitetty alla:

```
<OBJECT ID = "commercial"
        CLASSID="CLSID:CFCDAA03-8BE4-11cf-B84B-
        0020AFBBCCFA"
        WIDTH=240
        HEIGHT=160
```

```
<param                                                    name="SRC"
value="pnm://193.66.52.23/pics/jonoviesti2.ram">
<param name="CONTROLS" value="ImageWindow">
<param name="PLUGINSPAGE" value="http://www.real.com/">
</OBJECT>
```

Taulukko A.3: RealMedia-kontrollin parametrit

Parametri	Selite
ID	ID, jota muut HTML-sivulla olevat objektit käyttävät. Voi olla mielivaltainen – sivun sisällä yksikäsitteinen – merkkijono.
CLASSID	Globaalisti yksikäsitteinen ID, jota tietokone, ja muu internet käyttää juuri tämän luokan tunnistamiseen. Formaattina 256-bittinen kokonaisluku.
SRC	RealMedia-serverille esitettäväksi annettu RealMedia-tiedoston pnm-protokollan ja web-serverin hierarkian mukainen osoite.
CONTROLS	Määrittelee, mitkä osat RealMedia-COM-kontrollista asetetaan näkyviksi. ImageWindow näyttää pelkästään tiedoston ilman eri kontrolleja tai nappeja.
PLUGINSPAGE	Sen web-serverin URL, johon selain ohjataan menemään, mikäli asiakkaan ympäristössä ei ole rekisteröitynä RealPlayer COM–objektia.

Asennusvaiheessa ei tarvitse enää erikseen asentaa RealPlayer-objektia HTML-sivulle, kun se sinne on jo kerran kirjoitettu. RealMedia-serveri on kuitenkin installoitava, jollei sitä vielä ole valmiina.

1.1.5 Tiedostot

IPCC-projekti on toteutettu javalla. Käytetty kääntäjä oli Microsoft J++ 1.1:n mukana tullut versio, jonka pohjana käytetään JDK 1.0.2:ta. Projektiin kuuluvat tiedostot ovat:

- CliCtrl.dsp (MSVJ:n työtila-tiedosto)
- CliCtrl.java ja ajotiedosto CliCtrl.class, johon viitataan HTML-tiedostossa. (Sis. AXI ja käyttöliittymä)
- CliThread.java ja ajotiedosto CliThread.class (socket-yhteys)
- JMsg.java + class (kantaluokka alhaisen tason tietoliikenteeseen)
- CliMsg.java +class (JMsg-johdetut luokat erityyppisille viesteille)
- CCMSG.java + class (eräitä tyyppimäärittelyjä)
- NMDlg.java + class (asiakkaalle ilmoitettavista viesteistä vastaava dialogi)
- GroupItem.java + class (palveluryhmän luokka)
- GroupList.java + class (lista ryhmistä ja hakufunktioita)

- GroupTable.java + class (kokonaislukulista, käytetään mm. Määrittelemään esitetyt ryhmät)
- CliCtrl.html (kokoajana toimiva HTML-tiedosto)

1.2 MCU

1.2.1 iC-rajapinta ja sen tietoliikenneyhteydet

iC-rajapinta on se rajapinta, jonka lävitse MCU kommunikoi IPCC:n instanssien kanssa. Rajapinta on CMsg-luokkien päälle rakennettu viestipohjainen rajapinta, rakenteeltaan samanlainen CustomerConnectin X-rajapinnan kanssa. Kun MCU käynnistetään, viestijono aloittaa verkon kuuntelemisen tietyn IP-osoitteen ja määrätyn portin alla. (Kyseiset parametrit on määritelty MCU:n alustustiedostossa) Edempänä olevassa kappaleessa 1.2.4 esitetyn yhteydenluomiskäytännön mukaan viestijono hyväksyy yhteydenottopyyntöjä eri IPCC:n instansseilta, sekä erikoistapauksena myös IPX:n MCU:hun päin synkronisen yhteyden. Kun yhteys on syntynyt, sitä kuvaava socket talletetaan MCU:n sisäisiin rakenteisiin listan jäseneksi, jotta se on mahdollista löytää myöhemmin.

MCU:n kannalta viesti IPCC:lle lähetetään yksinkertaisesti konstruoimalla oikean CMsg-luokasta johdetun luokan instanssi ja lähettämällä se globaalista IPCC:n instanssit sisältävästä (yksi sisäisistä rakenteista, ks. 1.2.5) listasta haetulle client-luokan objektia edustavan socketin kautta. Lähetysfunktio on CMsg-kantaluokan jäsenfunktio. IPCC:tä vasten ei tueta synkronista lähetystapaa.

Viestijono-objekti vastaanottaa kaiken MCU:lle saapuvan datan, myös IPCC:ltä tulevat viestit. Se prosessoi ne kappaleessa 1.2.4 kuvatun käytännön mukaisesti, mutta tietoliikennekirjaston infrastruktuuria käyttävälle ohjelmoijalle riittää toteuttaa käytetyn CMsg-kantaluokan `Process()`-metodi.

iC-rajapintaan kuuluvat viestit on kuvattu alla, taulukossa A.4

Taulukko A.4: iC-rajapinnan viestit

Viesti	Suunta	Parametrit	Kommentit
IC_CALL_SERVICE	MCU ← IPCC	int GID, int CustID, String[] CustData	IPCC lähettää palvelupyynnön koskien tiettyä ryhmä-ID sisältäen mahdolliset tunnistetiedot
IC_CONTACTING	MCU → IPCC	int CallID	MCU informoi IPCC:lle palvelupyynnön

			statuksen: yhteydenotto CustomerConnectiin
IC_QUEUEING	MCU → IPCC	int CallID	MCU informoi IPCC:lle palvelupyynnön statuksen: yhteys luotu, palvelua odotetaan.
IC_MAKE_CALL	MCU → IPCC	int CallID, String IP-address	MCU komentaa IPCC:n aloittamaan puhelun määrättyyn IP-osoitteeseen
IC_CALL_HUNGUP	MCU ← IPCC	int CallID	IPCC informoi MCUta manuaalisesti katkaistusta puhelusta.
IC_HANGUP_CALL	MCU → IPCC	int CallID	MCU komentaa IPCC:n katkaisemaan käynnissäolevan puhelun (jos edelleen käynnissä).
IC_LOGIN	MCU ← IPCC	String IP-address	IPCC suorittaa login-tunnistuksen MCU:lle.
IC_REQUEST_GROUPS	MCU ← IPCC	int Cuid	IPCC pyytää MCUlta listaa saatavilla olevista palveluista.
IC_GROUPS_RCVD	MCU → IPCC	int iSize, GroupList *pGroupLst	MCU lähettää ryhmä/palvelu-tiedot IPCC:lle.
IC_ERROR	MCU → IPCC	int iType, String sReason	Jos pyyntöä ei voida täyttää, MCU vastaa tällä viestillä.
IC_DBG_NOTIF	MCU ← IPCC	String Log-string	Debug tarkoituksiin oleva viesti (käytetään vain, jos tuote testausvaiheessa).

1.2.2 I-rajapinta ja sen tietoliikenneyhteydet

Yhteydet tästä MCU:sta CustomerConnectin IPX-serveriin kulkevat I-viestirajapinnan lävitse. Se on rakenteeltaan ja infraltaan kuten iC-rajapintakin, ja sijainniltaan vastaa esim CustomerConnectin TAPI SP:n viestirajapintaa. Rajapinta koostuu useista viesteistä ja yhdestä kolmen fyysisen socketin muodostamasta loogisesta yhteydestä. Mikäli IPX (eli CustomerConnect-serverin ajotiedosto) ei ole käynnissä, MCU on melko toimintakyvytön tämän työn tarkoittamassa järjestelmässä. Jos MCU:n käynnistäminen ilman IPX:n olemassaoloa olisi mahdotonta, ei olisi mahdollista käynnistää koko IP-puolta järjestelmästä. Tämän johtuu CSTA-mallin pohjalle rakennetun perusjärjestelmän yhdestä perusoletuksesta: tiettyä mediaa käsittelevä vaihdekomponentti on saatavilla silloin, kun CustomerConnect käynnistyy. Niinpä vaikka MCU:n alustus aluksi epäonnistuukin, se ei silti sulje itseään, vaan jatkaa kytkeytymisyrityksiään IPX:ään, kunnes prosessi lopetetaan.

Yhteys IPX:ään koostuu kolmesta erillisestä socketista, jotka on kuitenkin loogisesti sidottu yhdeksi nipuksi. Syy tähän on tietoliikennekirjaston asynkroninen rakenne: asynkroniseksi määritellyn socketin lävitse on mahdollista kommunikoida molempiin suuntiin, mutta vain asynkronisesti. Synkronisilla yhteyksillä lähettävän osapuolen – sen, jonka säikeen suoritus keskeytetään, kunnes viestiin saapuu vastaus tai sen aikaraja tulee vastaan – on luotava asynkronisesta socketista erillinen socket sille osapuolelle, jolle synkroninen kutsu lähetetään. Vastaanottava puoli ei kuitenkaan pysty käyttämään tätä sockettia muuhun kuin vastausten lähettämiseen synkroniseen kutsuun. Se ei pysty lähettämään tavallisia asynkronisia viestejä tai omia synkronisisa viestejään tämän kanavan lävitse, joskin vastaanottava osapuoli ei erota sille päin luotua synkronista ja vakiona olevaa asynkronista kanavaa toisistaan muuten kuin jäsenmuuttujan nimestä.

Jotta siis pystyttäisiin kommunikoimaan sekä synkronisesti että asynkronisesti molempiin suuntiin, molempien osapuolten - sekä IPX:n että MCU:n - täytyy luoda omat synkroniset kanvansa. Yhdessä nämä mahdollistavat sekä RPC:n kaltaiset että asynkroniset tapahtumaohjatut yhteydet kumpaankin suuntaan.

MCU:lta IPX:lle päin oleva synkroninen yhteys (MCU on synkronisten kutsujen lähettäjänä) on poikkeus sille säännölle, että jokainen viesti kulkee viestijonon lävitse: tässä tapauksessa lähettävä säie odottaa, kunnes vastaus on saapunut, eikä viestin `Process()`-metodia kutsuta. Tässä tapauksessa alemman tason tietoliikennefunktiot pitävät huolen vastausviestin toimittamisesta lähettävälle säikeelle. (ks. kuva A.8 osiossa 1.2.3)

I-rajapinta on iC-rajapinnan kanssa sikäli samanlainen, että molemmissa alemman tason tietoliikennekomennot on abstrahoitu pois CMsg-luokan taakse. Lähettäminen tapahtuu luomalla sopiva CMsg- johdetun luokan objekti ja kutsumalla sen send()-metodia. Oikean kanavan valitsemista varten on komennettava viestinkantaluokan SetSocket()-funktiota parametrinään joko asynkronisen tai synkronisen kanavan socket. Kaikki saapuvat viestit lukuunottamatta synkronisiin pyyntöihin vastauksina saatuja viestejä kiertävät viestijono-objektin kautta.

I-rajapintaan kuuluvat viestit on esitetty taulukossa A.5.

Taulukko A.5: I-rajapinnan viestit

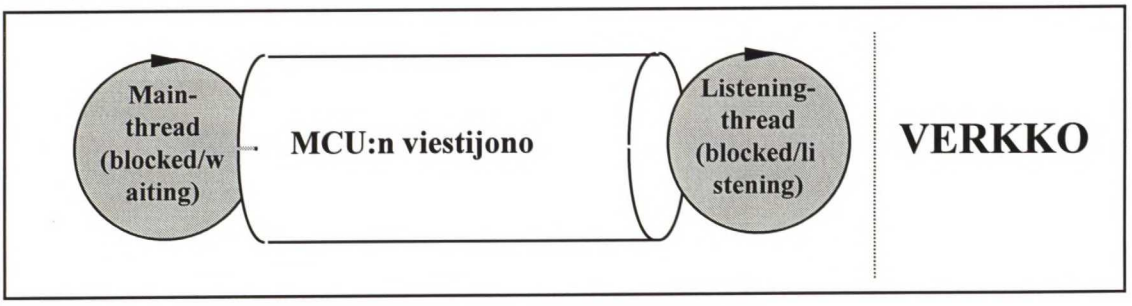
Viesti	Suunta	Parametrit	Kommentit
IS_MAKE	MCU ← IPX	Puhelu-ID, B-Numero, Numeron tyyppi, B-numeron ID (ei käytössä)	IPX komentaa IPCC:n aloittamaan puhelun
IS_HANGUP_CALL	MCU ← IPX	Asiakkaan laite-ID	IPX komentaa IPCC:tä katkaisemaan puhelun
IS_HANGUP_CALL_ACK	MCU → IPX	Asiakkaan laite-ID, paluukoodi (eRetCode)	Vahvistusviesti ylläolevalle viestille
IS_DIVERT_CALL	MCU ← IPX	Asiakkaan laite-ID, C-numeron laite-ID ja lisätietoja	IPX kertoo IPCC:lle kutsunsiirrosta.
IS_DIVERT_CALL_ACK	MCU → IPX	Paluuarvo divertointipyyntöön, uusi B-numero ja puhelun tila (eCallState)	Vahvistusviesti ylläolevalle viestille
IS_LOGIN	MCU → IPX	Käyttäjätunnus, salasana, IP-osoite, -portti, A-numero ja synkronisen liikennöinnin tyyppi (ei käytössä)	MCU kirjautuu IPX:ään.
IS_LOGIN_ACK	MCU ← IPX	Käyttäjän ID / käyttö kielletty	IPX vahvistaa kirjautumisen
IS_FETCH_GROUPS	MCU → IPX	Asiakkaan ID	MCU hakee saatavilla olevat palveluryhmät CustomerConnectin tietokannasta
IS_FETCH_GROUPS_ACK	MCU ← IPX	Ryhmähakupyynnön paluuarvo ja	Vahvistusviesti edelliseen ja sen

		ryhmädata.	mukana tuleva ryhmädata
IS_CALL_HUNGUP	MCU → IPX	Katkaistun puhelun laite-ID	MCU notifies IPX of a IPCC-hangup
IS_CALL_STATE(ei käytössä)	MCU → IPX	Laite-ID, agentin ID, puhelun tila (CallState), reititystiedot + Huomautuskenttä + clientissa kiinni olevan agentin ID (ei käytössä)	MCU informoi IPX:lle muuttuneen puhelun tilan (käytössä, kun mediavirta voidaan laittaa pitoon, yms,)
IS_INCOMING_CALL	MCU → IPX	Soitetun numeron laite-ID ja Customer Info-rakenne (asiakkaan tunnistetiedot).	MCU ilmoittaa IPX:lle tulevasta puhelusta.
IS_REQUEST_SERVICES	MCU → IPX	-	Varattu myöhempään käyttöön.
IS_REQUEST_SERVICES_ACK	IPX ← MCU	-	Varattu myöhempään käyttöön.

1.2.3 MCU:n viestijono

MCUssa on vain yksi sisääntulevia I- ja iC-rajapinnan viestejä synkronoiva viestijono. Tämän viestijonon muuttujanimi on G_pServerQueue (CustomerConnectin tietoliikennekirjastojen tekemien oletuksien pakottamana). Viestijono on FIFO-tyyppinen jonokuria noudattava objekti, jonka WinMain-säie käynnistää MCU:n käynnistyksen yhteydessä. Viestijonoja on MCUssa vain yksi instanssi, ja sen käyttöön perustuu koko MCU-sovelluksen synkronointi. Viestijonon ja CustomerConnectin tietoliikennekirjastojen yksityiskohtainen funktiotason toiminta on esitetty *CustomerConnect, Technical Documentation. Version 2.0.* – dokumentissa. Tässä annetaan vain säietason kuvallinen esitys viestijonosta ja sen toiminnasta. Säikeiden nimistä ja niiden tiloista käytetään näiden englanninkielisiä nimiä itse koodissa olevien merkintöjen ymmärtämisen helpottamiseksi.

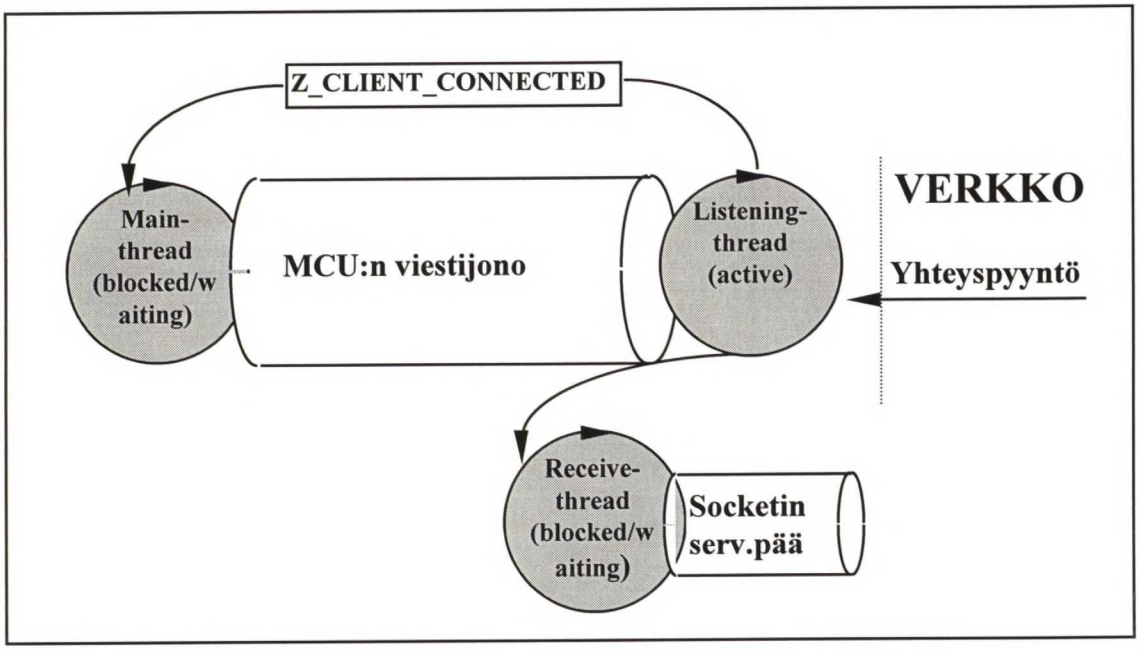
Tyhjä viestijono (MCU ilman yhteyksiä) ei tee muuta kuin kuuntelee verkkoa odottaen IPCC:tä tai IPX:ää kytkeytyväksi.



Kuva A.1: MCU:n viestijono ja siihen liittyvät säikeet alkutilassa (ei saapuvia tai prosessoitavia viestejä)

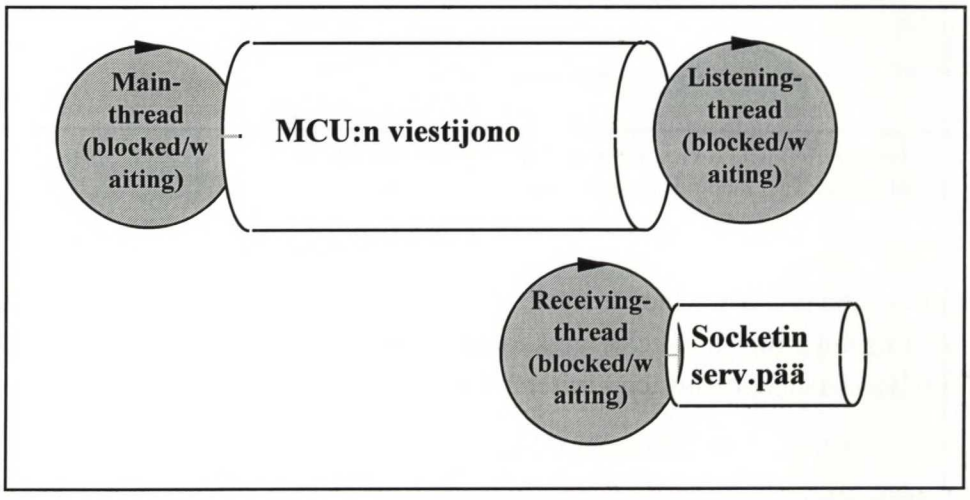
Kun viestijono luodaan, sen yhteydessä luodaan kaksi säiettä: nk. listening-thread ja main-thread. Ensinmainittu kuuntelee verkkoa uusien yhteydenottojen varalta, kun taas viimemainittu tarkistaa jonoa uusien käsiteltävien viestien varalta.

IPCC:n kytkeytyessä luodaan uusi socket, jolle asetetaan vartija-säie, nk. Receive-thread odottamaan tästä nimenomaisesta socketista tulevia viestejä. Korkeampien abstraktiotasojen komponentteja informoidaan myös tässä vaiheessa uudesta kytkeytyneestä clientista.

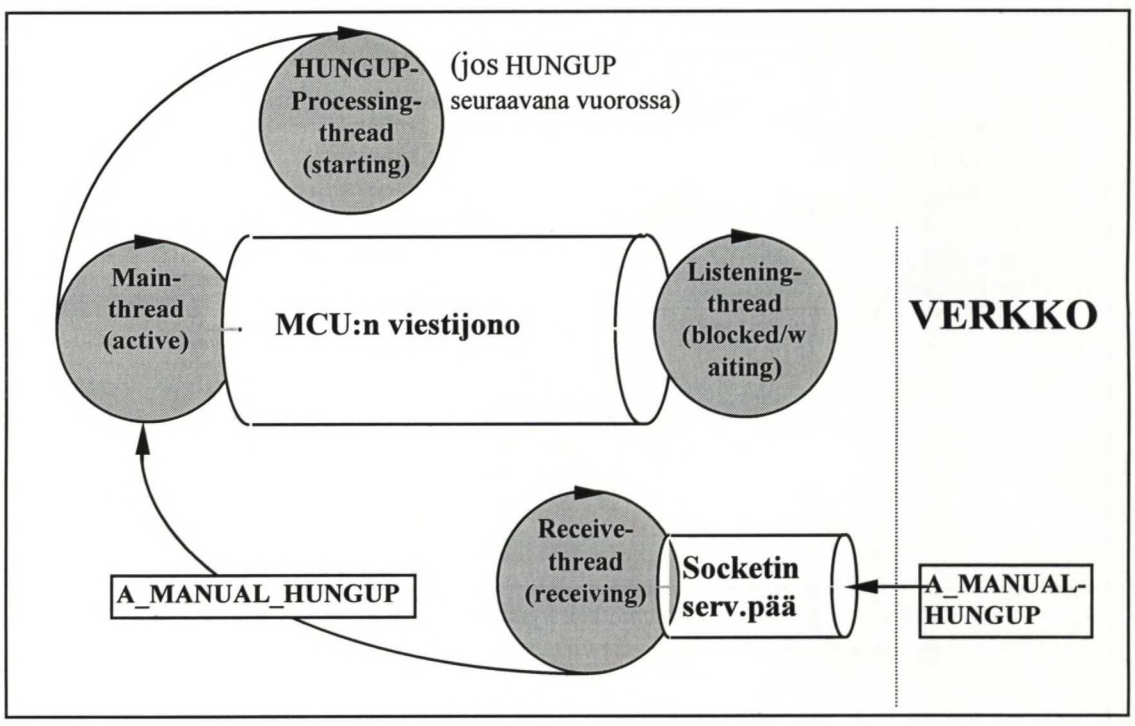


Kuva A.2: MCU:n viestijono:client kytkeytyy. Listen-thread käsittelee pyynnön ja lähettää siitä viestin viestijonolle

Kun asynkroninen viesti saapuu, (tämä pätee kaikkiin MCU:ssa ilmeneviin tapauksiin, paitsi milloin odotetaan vastausta synkroniseen viestiin) viestijono siirtyy **IDLE**-tilasta **RECEIVING**-tilaan. Tässä viesti siirretään tietoliikennekerrokselta sovelluskerroksen käsiteltäväksi, jossa viestijonon main-thread välittömästi synnyttää uuden säikeen, Processing Threadin, käsittelemään viestiä.

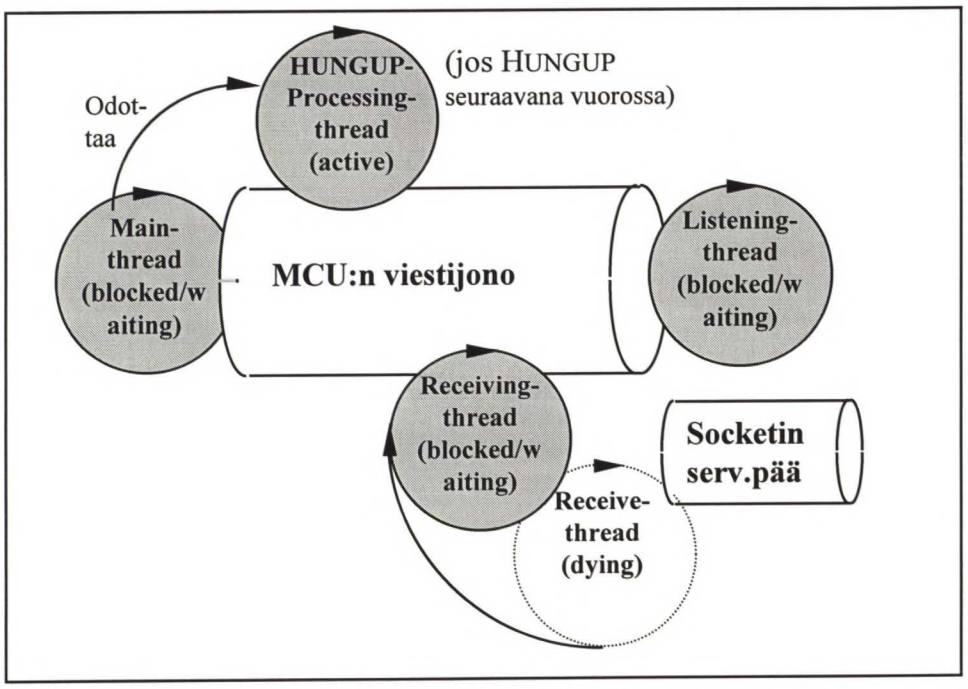


Kuva A.3: MCU:n viestijono: **IDLE**-tila. Client kytkeytynyt, muttei lähetä viestejä.



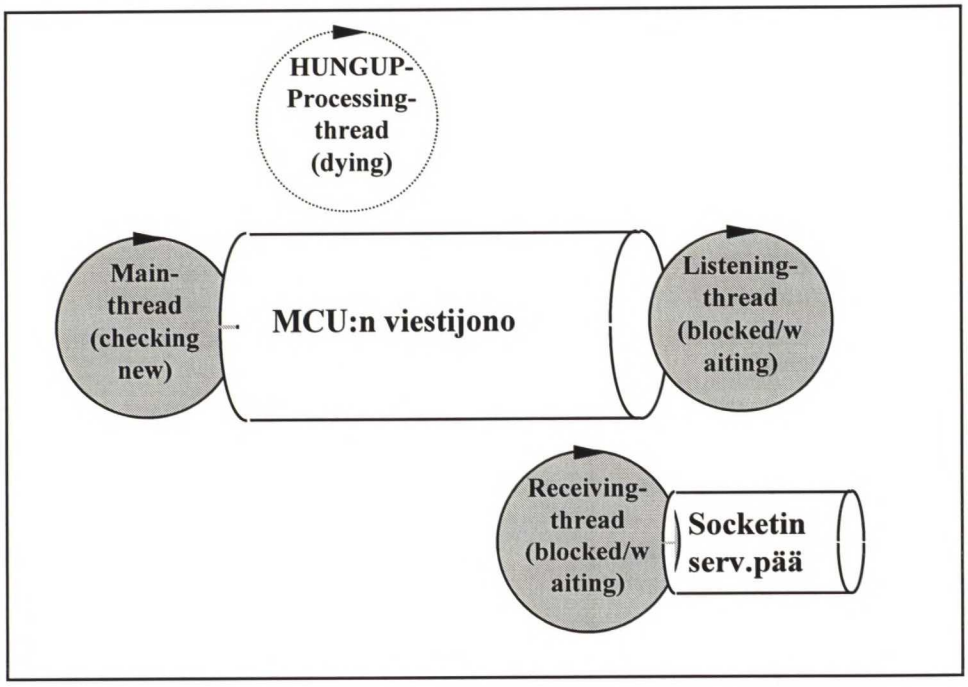
Kuva A.4: MCU:n viestojono: uuden viestin käsittely (1/3)

Kun viestin käsittely on aloitettu, Receive-thread palaa idle-tilaan tuhoamalla itsensä ja luomalla uuden Receive-threadin. Main-thread odottaa, kunnes Processing-thread ilmoittaa käsitelleensä viestinsä antaakseen jonorakenteen uusien viestien käyttöön.



Kuva A.5: MCU:n viestijono: uuden viestin käsittely (2/3)

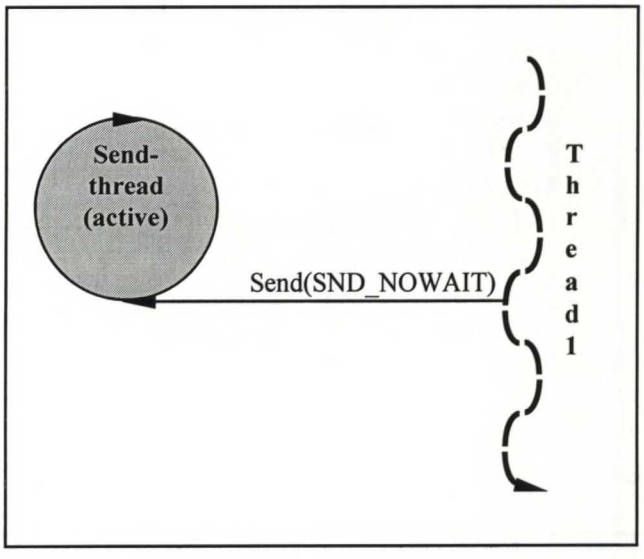
Kun prosessoiva säie on joko lopettanut suorituksensa kokonaan, tai vähintään kriittisten tehtävien suorituksen, pääsäie tarkistaa jonossa mahdollisesti olevien muiden viestien olemassaolon, mutta muuten järjestelmä on taas IDLE-tilassa.



Kuva A.6: MCU:n viestijono: uuden viestin käsittely päättynyt (3/3). Palaa IDLE-tilaan, jos jonorakenteessa ei ole enää uusia prosessoitavia viestejä.

Ainoa vastaanotettavan viestin tyyppi, joka ei kulje viestijonon kautta, on synkroniseen pyyntöön vastauksena tuleva viesti. Tämän ymmärtämiseksi on ensin tarkasteltava

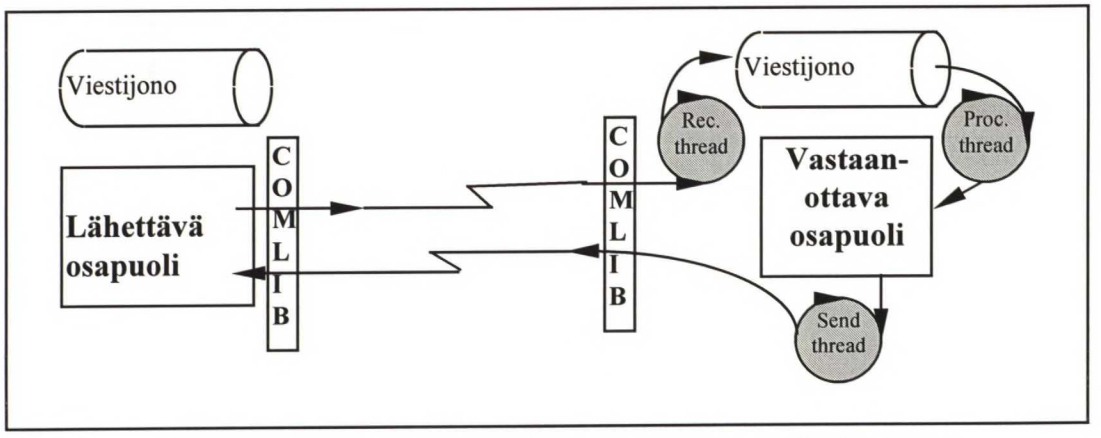
viestin lähetysrutiinia. Kun lähetetään viesti, konstruoidaan normaalisti ensin sopivantyyppisen viestiluokan instanssi, jonka jälkeen kutsutaan vietin Send()-metodia parametrin arvolla SND_NOWAIT. Tämä saa aikaan uuden nk. lähettäjäsäikeen (Send-thread) luomisen, joka lähettää viestin asynkronisesti.(Kuva A.7)



Kuva A.7: Asynkroninen viestinlähetys: suorituksessa oleva säie luo uuden säikeen pelkästään viestin lähettämistä varten.

Synkronisen viestin tapauksessa ei kuitenkaan ole olemassa sovellustason versiota asynkronisen lähetyksen kaltaisesta Send()-metodista. (Esim. vastaanottavat funktiot osoittimineen on konstruoitava erillään tietoliikennekerroksen funktioista.) Tietoliikennekirjastojen sallima tapa tehdä synkroninen lähetys on käyttää kirjastoja suoraan, ilman viestijonoa. On huomattava, että ainoastaan lähettävä osapuoli näkee toiminnan synkronisena: vastaanottava osapuoli ei pysty erottamaan synkronisesti ja asynkronisesti lähetettyjen viestien välillä, koska molemmat niistä saapuvat saman viestijonon kautta. Ainoastaan vastaus synkroniseen pyyntöön ei saavu lähettävän osapuolen viestijonon kautta. Sen sijaan lähettäjän send()-funktiokutsu pysäyttää sitä kutsuneen säikeen suorituksen, kunnes se saa joko vastauksen viestiinsä tai ylittää odotukselle asetetun aikarajan. (Ks. kuva A.8)

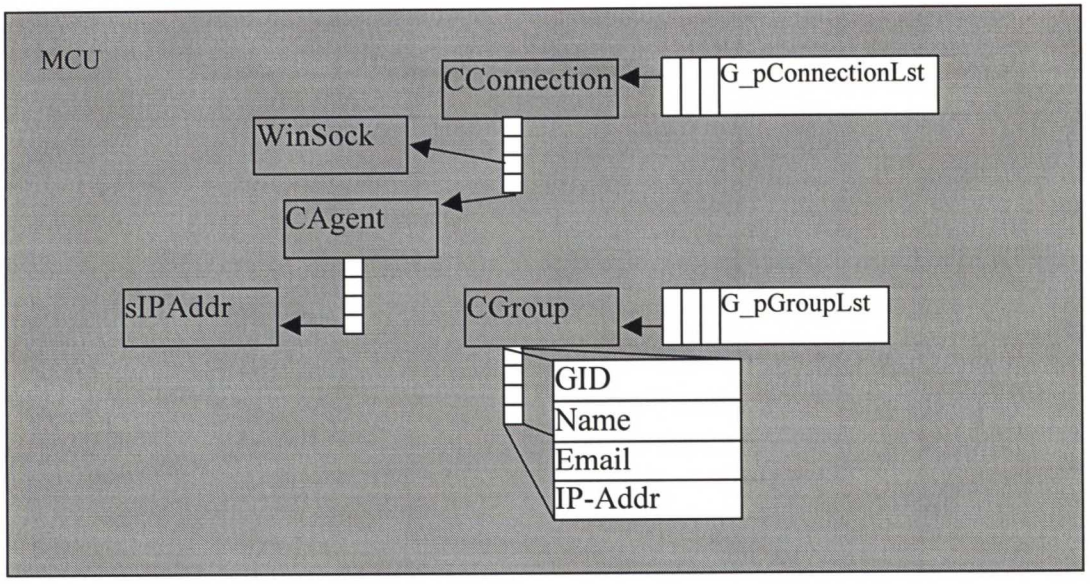
Edellä esitetyissä kuvissa useita rinnakkaisia säikeitä suorittaa yhtäaikaa. Receive-Listening ja Main-säikeet eivät kuitenkaan käytä muita yhteisiä tietorakenteita kuin viestijonoa, joten muuhun kuin jonorakenteeseen liittyvään synkronointiin ei toistaiseksi ole tarvetta. Toisaalta Processing-säikeiden koodista osa sijaitsee kirjastoa käyttävän ohjelmoijan toteuttamassa Process()-metodissa, joten Processing-säikeiden ei voi antaa suorittaa mielivaltaisesti, joten ne on sarjallistettu sallimalla vain yhden prosessoivan säikeen suorittaa kerrallaan.



Kuva A.8: Synkroninen viestinlähetys: Vastaanottava puoli näkee tämän normaalina asynkronisena viestinä. Lähettävällä puolella ei kuitenkaan luoda uusia säikeitä, ja viestit välitetään suoraan tietoliikennekirjastojen funktioiden avulla. Erityisesti huomattava, että viestijonoon ei lähettävällä puolella kosketa kertaakaan.

1.2.4 Tärkeimmät sisäiset rakenteet

IPX:n ja IPCC:n välisen tulkinnan toteuttamiseksi MCU:ssa on joitakin listoja, joiden avulla suoritetaan kuvaus IPX:n CSTA-keskeisestä maailmasta IPCC:n H.323-keskeiseen maailmaan ja päinvastoin. Listat sisältävät ryhmälistan ja listan "yhteyksistä", joka sisältää kytkeytyneiden IPCC:den osoitteet ja muuta dataa. Sisäiset rakenteet on esitetty alla, kuvassa A.9.



Kuva A.9 MCU:n sis. rakenteet

1.2.5 IADM-käyttöliittymä

Yksinkertaista paikallista ylläpitoa ja tietyntasoiseen testaukseen tarkoitettu käyttöliittymä IADM sisältää funktiot mm. MCU:n pysäyttämiseen ja sammuttamiseen ja testiviestien lähetykseen valitulle IPCC:lle. (MCU:n käynnistys- ja sulkemistoiminnot on siirretty itse sovelluksen käynnistämisen ja sulkemisen yhteyteen, joten itse menu-

toiminnoilla ei ole enää merkitystä.) Testiviesteillä voi tarvittaessa simuloida erilaisia virhetilanteita ilman CustomerConnecia ja tehdä huomioita IPCC:n käyttäytymisestä. Testiviestien lähetykseen tarkoitettu dialogi on asetettu erään päävalikossa olevan menun alle.

1.2.6 Asennus

MCU koostuu vain yhdestä ajotiedostosta (.EXE-tiedosto), jonka voi sijoittaa Windows NT Server 4.0 käyttöjärjestelmän alla toimivaan tietokoneeseen. Systeemihakemistossa on alustustiedosto `mcu.ini`, jonka asetukset ja parametrit ovat seuraavat:

- MCU-address & MCU-port: se portti ja IP-osoite jota MCU:n viestijonon pitäisi kuunnella
- IPX-address & IPX-port: IPX-serverin IP-osoite ja portti. MCU yrittää kytkeytyä tässä osoitteessa olevaan viestijonoon.
- IPX-pollinginterval (sekunteja): aikavakio, joka kertoo kauanko MCU odottaa ennen kytkeytymistään uudelleen IPX:ään yhteyden menettämisen tai luomisen epäonnistumisen jälkeen.
- Demo_groups_generate: debug- ja demo-tarkoituksiin IPX:n ei tarvitse välttämättä olla olemassa ryhmätietojen hakua varten. Jos tämä parametri asetetaan ykköseksi, tuotetaan tekaistuja ryhmiä.
- Fake_agent_IP: jollei tätä jätetä tyhjäksi, sen ilmoittamaa IP-osoitetta käytetään antamaan vaste **IC_CALL_SERVICE**-kyselyyn välittömästi. (IPCC:lle kerrotaan – virheellisesti – että agentti annetussa IP-osoitteessa on valmi vastaanottamaan puhelun) Myös tätä käytetään vain debug- ja demo-tarkoituksiin.

1.2.7 Tiedostot

Projekti on tehty Microsoftin kehitysympäristössä, tarkemmin sanoen Microsoft Visual C++ 4.2:lla, käyttöjärjestelmänä NT 4.0 server. Projektiin kuuluvat tiedostot:

- InterSrv.mdp (työtila- ja projektitiedosto)
- InterSrv.cpp (sisältää soveluksen pääluokan ja käynnistysrutiinit)
- IntMsg.cpp (sisältää I- ja iC-rajapinnat)
- Intsockadmd.cpp (yksi IADMin UI-tiedostoista)
- IntConn.cpp (sisältää MCU:n sisäisen kuvan IPCC:stä)
- IntAge.cpp (sis. agentti / asiakas - luokat)
- InterSrvView.cpp (yksi IADMin UI-tiedostoista)
- InterSrvDoc.cpp (yksi IADMin UI-tiedostoista)

- IPXgro.cpp (IPX-ryhmän luokka on sisällytettävä projektiin **IS_FETCH_GROUPS**-viestien vuoksi)
- cbsinf.cpp (CustomerInfo tietorakenne, sis. tunnistetietokentät)

1.3 IP-vaihdeserveri

1.3.1 I-rajapinta ja sen tietoliikenneyhteydet

Itse I-rajapinta ja sen sisältämät viestit on kuvattu jakson 1.2.3 yhteydessä. Koska MCU on tarkoitetussa hierarkiassa IPX:ää alempana, yhteen IPX:ään voi kytkeytyä useampi MCU (joskin tällä hetkellä sallitaan vain yksi MCU CustomerConnectin tilaserveriä kohden). Tämä tarkoittaa useampia kolmen socketin muodostamia loogisia yhteyksiä samalle IPX:lle, kukin yhteys sisältäen täydet synkroniset/asynkroniset kanavat viestinlähetykseen. IPX:n kannalta MCU:n socketit ovat ainoa yhteys ulkomaailmaan: kommunikointi iX-rajapinnan lävitse CustomerConnectin tilaserveriä vasten tapahtuu saman prosessin sisällä yhteisessä muistiavaruudessa.

MCU-socketit luodaan vasta MCU:n kytkeytyessä, joten IPX:n on mahdollista käynnistyä, vaikka se *ei löytäisikään vaihdekomponenttia*. Luonnollisesti yksikään IP-puhelu ei voi tällöin kulkea järjestelmän kautta, mutta IPX ei saa kaatua tai kieltäytyä käynnistymästä olemattoman MCU:n vuoksi. Edellämainittu ominaisuus on erityisen tärkeä siksikin, että IPX on osa koko CustomerConnectin serveriä, vaikka onkin vain yksi vaihdeservereistä. MCU:ta voi myös olla olemassa monta instanssia yhtäaikaa, ja ne voivat kytkeytyä toistaiseksi tuntemattomista paikoista: IPX ei edes tiedä, mistä hakea jotakin tiettyä MCU:ta, ja toisaalta virhe IP-puheluiden käsittelyssä ei saa lamauttaa PSTN-puheluja. Jokaisen MCU:n on myös pystyttävä kytkeytymään ja poistumaan mielivaltaisesti, koska yhdenkään yksittäisen MCU:n ei voi vaatia olevan koko ajan CustomerConnectin käytettävissä.

Vaihdekomponentin puuttuminen ei aiheuta ongelmia: jos MCU ei ole ylhäällä, yksikään IPCC:tä käyttävä agentti tai asiakas ei pysty kytkeytymään Customer-Connectiin. IPCC:n puuttuminen eliminoi inbound-puhelut ja ne outbound-puhelut, jotka kuuluvat CustomerConnectin alueelle (Jos joku aloittaa puhelun pelkän IP-puheluohjelman avulla, sen otaksutaan olevan henkilökohtainen CustomerConnectin ulkopuolella oleva puhelu.) Toisaalta MCU ei saa kytkeytyä pois järjestelmästä niin kauan kuin yksikin IPCC on kytkeytyneenä siihen. Ainoa tapa kytkeä MCU irti IPX:stä on lopettaa MCU:n prosessi joko käyttöliittymästä tai NT:n prosessinhallinnasta. Kumpikin tapaus on verrattavissa virtakatkokseen tai verkon häiriötilaan, eli kyseessä on *force majeure*. Tällöin MCU:ta ei pidä pysäyttää kuin esim. huoltotöiden vuoksi.

Jotkin viestit ovat luonteeltaan synkronisia ja toiset taas asynkronisia. Synkroniset viestit lähetetään kunkin kyseessä olevan MCU:n synkronista sockettia pitkin, ja vastaus tulee samaa kanavaa myöten. Kaikissa muissa tapauksissa mahdolliseen pyyntöön

annettu vastaus lähetetään samaa kanavaa myöten kuin pyyntökin saapui, tai jos kyseessä on ilmoitusluontoinen tapahtuma (event), yleistä asynkronista kanavaa myöten. Kaikki tulevat viestit, poislukien synkroniseen pyyntöön saatu vastaus, kulkevat IPX:n viestijonon kautta. (Kuvattu luvussa 1.3.3)

1.3.2 iX-rajapinta ja sen tietoliikenneyhteydet

iX-rajapinta on itse asiassa X-rajapinnasta muokattu osajoukko, mikä tarkoittaa, että vain osa X-rajapinnan viesteistä on toteutettu tässä rajapinnassa, ja että toteutettujen viestien parametreissä ja prosessointimetodien toiminnassa on joitakin eroja. iX-viestit ovat kaikki asynkronisia, vaikka ne saattavatkin olla loogisesti synkronisia. Täten kaikki tämän rajapinnan lävitse kulkevat viestit laitetaan joko suoraan tilaserverin viestijonoon, tai haetaan ne IPX:n viestijonosta.

IPX kääntää iX-viestit IS-viesteiksi, ja lähettää ne edelleen MCU:lle, muttei kuitenkaan aivan sokeasti. Joitakin rationalisointeja tehdään PSTN-kytkimen osaan sovelletun tilaserverin ja MCU:n käyttämien mallien välillä. Kaikkia viestejä ei lähetetä eteenpäin, koska IP-puhelut ja PSTN-puhelut eivät ole suoraan vertailukelpoisia, ja IP-puheluiden käsittely vaatii mediavirtaa käsittelevän ohjelmiston komentamista, jossa taas on omat rajoituksensa. Nämä kaksi faktaa sanelevat pääasiassa IP-maailmassa toistaiseksi toteutettavissa olevat CSTA-operaatiot. Esimerkiksi puheluita ei varsinaisesti jonoteta, vaikka se tilaserverille ja puhelua tekevälle asiakkaalle kylläkin näyttää jonotukselta. Tällöin, kun tilaserveri määrää siirtämään yhteyden ryhmän loogisesta osoitteesta jonotusta tarjoavaan osoitteeseen, IPX ei tee mitään muuta kuin lähettää vastausviestin tilaserverille. Vastaavasti varsinaista mediavirtaa ei aloiteta, ennenkuin tilaserveri käskee vaihteen vastata hälyttävään puheluun. Yksityiskohtaisempi selostus annetaan toiminnallisista kuvauksista kertovassa luvussa.

iX-viestijoukko on kuvattu allaolevassa taulukossa A.6.

Taulukko A.6: iX-rajapinnan viestit

Viesti	Suunta	Parametrit	Kommentit
X_ALERTING	SS ← IPX	int ConnID, Cstring DeviceIDA, DeviceIDB, DeviceIDOrig, CallerID, int OptParam	MCU:n ilmoitus inbound-puhelusta tilaserverin suuntaan.
X_DIVERT-CALL	SS → IPX	int ConnID, Cstring DeviceIDB, DeviceIDC, DeviceIDExt	Tilaserveri komentaa hälyttävälle puhelulle kutsunsiirron.
X_DIVERT-CALL_CONF	SS ← IPX	int ConnIDB, ConnIDC, enum RetCode	Vahvistusviesti X_DIVERT_CALLille
X_ANSWER-	SS → IPX	int ConnIDB	Tilaserveri komentaa

CALL			hälyttävän puhelun vastaamaan.
X_ANSWER-CALL_CONF	SS ← IPX	int ConnIDB, enum RetCode	Vahvistusviesti X_ANSWER_CALLille
X_HANGUP	SS → IPX	int ConnID	Tilaserveri komentaa IPX:ää katkaisemaan jonkin käynnissä olevan puhelun.
X_HANGUP-CONF	SS ← IPX	int ConnID, enum RetCode	Vahvistusviesti X_HANGUPille
X_MAKE_CALL	SS → IPX	int ConnIDA, Cstring DeviceIDA, DeviceIDB, DeviceIDBExt, enum CallType	Tilaserveri komentaa IPXää aloittaman uuden outbound-puhelun.
X_INITIATED	SS ← IPX	Cstring DeviceIDA, int ConnID, enum RetCode	Vahvistusviesti X_MAKE_CALLille
X_MANUALLY_ HUNGUP	SS ← IPX	int ConnID	MCUn ilmoitus asiakkaan IPCC:stä tai IP-puheluohjelmasta käsin tai agentin IP-puheluohjelmasta käsin katkaisemasta puhelusta tilaserverille.

1.3.3 Viestijono

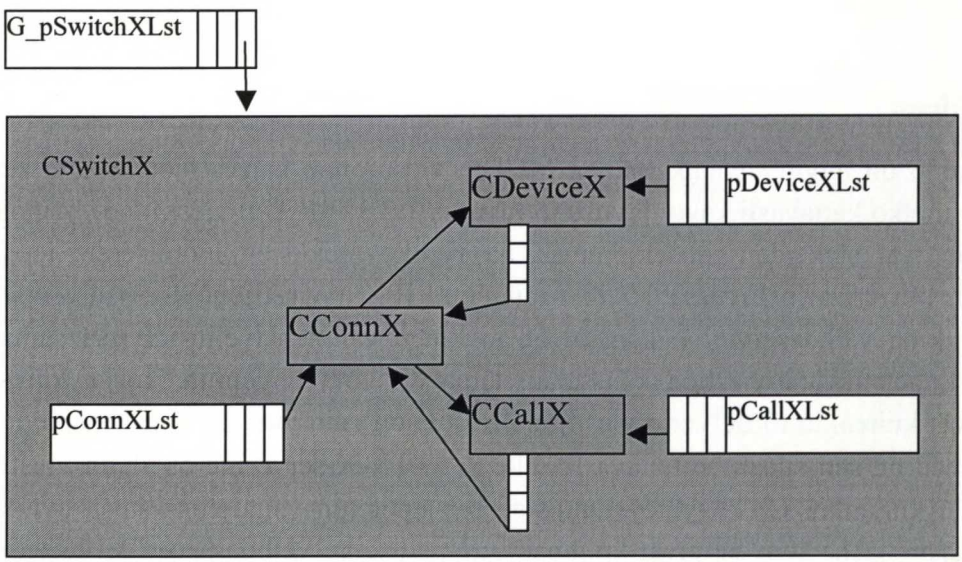
IPX:n viestijono on hyvin samankaltainen MCU:n viestijonon kanssa, joten se ja sen toiminnot on melko kattavasti käyty lävitse jo luvussa 1.2.3 (MCU:n viestijono). Ainoat isommat erot ovat säikeiden prosessoinnissa ja verkkoyhteyksien luomisessa, joista jälkimmäinen kuvattiin jo luvussa 1.3.1: I-rajapinta. IPX:n viestijono on itse asiassa yhteinen kaikkien vaihdeserverien kanssa, ts. jokainen vaihdepalvelin käyttää samaa viestijonoa. Lähetettävien viestien kanssahan tämä ei tuota ongelmia, koska kaikki veistit menevät kuitenkin tilaserverin viestijonolle, jota on vain yksi kappale. Saapuvien viestien suhteen on sen sijaan erotettava IPX:lle ja PSTN-X-serverille osoitetut viestit. Tämä tehdään kuhunkin CSTA-yhteystunnisteeseen upotetun vaihdetunnisteen avulla, ts. viestejä käsitellään, kuin ne eivät kuuluisi kummallekaan vaihdeserverille, ja vasta kun on nojauduttava vaihdeoperaatioita vaativiin toimenpiteisiin, kuten luotaessa CSTA-objekteja, ja käytettäessä niiden jäsenfunktioita, eriytetään eri vaihde- ja

mediatyypeille menevät viestit. IS-viestit ovat eri lukunsa, koska siellä samaa viestin tunnistetta ei käytetä kahdelle eri vaihteelle lähetetyille viesteille.

Vaihdeserverien viestijonossa viestejä prosessoivien säikeiden sallitaan suorittaa näennäisesti yhtäaikaa, mikä tarkoittaa, että vaikkakin vain yhden säikeen kerrallaan annetaan suorittaa, useita säikeitä voi silti olla samaan aikaan olemassa, vaikkakin pysähtyneenä odottamassa jotakin resurssia. Kun säie jää odottamaan jotakin resurssia, se voi eksplisiittisesti luovuttaa suoritusoikeuden muille säikeille. On huomattava, että lomitus tehdään viestipohjaisesti, eikä yksittäisten kriittisten sektioiden tasolla. Jokaisen viestin on suorituksensa alussa ilmoitettava, mitä (CSTA-) objekteja se aikoo käyttää, ja lukitukset tehdään ainoastaan ilmoitetuille objekteille siksi aikaa, kun viestin käsittelysäie suorittaa. Objektit pysyvät lukittuina, kunnes suorittava säie on lopettanut. Kriittisiä sektioita ei käytetä: vaikkakin tämä mahdollistaa suorituskyvyn optimoinnin, se vaikuttaa myös lähdekoodin luotettavuuteen (koska ohjelmiston suorituskyky ei ole järjestelmän pullonkaula niin kauan kuin PBX:ien vasteet ovat kymmensosasekunneista sekuntiin, luotettavuus on tärkein tavoite tämän järjestelmän puitteissa). Säikeiden lomitus tehdään tietoliikennekirjastoissa ohjelmallisesti, joten vaikka prosessori ja käyttöjärjestelmä kykenisivätkin vaadittuun moniprosessointiin, niiden ei anneta sitä tehdä.

1.3.4 Sisäiset rakenteet

Sisäiset rakenteet ovat CSTA-mallin mukaisia. Ne ovat olennaisilta osiltaan samoja, kuin muissakin vaihdeservereissä, ainoastaa jäsenfunktioiden toteutuksissa on eroja. CSTA-rakenteet esitetään tässä, koska ne ovat suhteellisen olennainen osa IPX:ää, ja koska koko CustomerConnectin teknistä dokumenttia ei tähän työhön ole tarkoituksenmukaista upottaa.



Kuvassa A.1.0 nuolet esittävät osoitteista (pointtereita): pointteri on jäsenmuuttuja siinä luokassa, josta nuolen kanta lähtee, ja pointterin sisältämä instanssi on nuolen kärjen

osoittamaa luokkaa. Ketjutetut suorakaiteet esittävät listoja, tässä tapauksessa pointtereiden listoja.

CSTA-mallissa jokainen fyysinen puhelu muodostuu kahdesta tai useammasta laitteesta (puhelinkone, muistipaikka vaihteessa, tai virtuaalinen puheluihin pystyvä ohjelmisto), puhelusta ja näiden välisistä yhteyksiksi kutsutuista assosiaatioista. (Huom: yhteys on assosiaatio täsmälleen yhden puhelun ja laitteen välillä, vaikkakin joissakin tapauksissa puhelu ja/tai laite voivat olla NULL-tilassa. Eri objekteilla voi vuorostaan olla eri tiloja. Näistä enemmän luvussa diplomityön luvussa 3.2, ja *CSTA Phase II Standard ECMA-217.)*

Ylläkuvattujen objektien esittämä CSTA-mallin mukainen kuva vaihteessa olevien puhelujen tilasta tulisi olla yhdenmukainen CustomerConnectin tilaserverin käsityksen kanssa: jokaiselle yhteydelle, laitteelle ja puhelulle on löydyttävä vastaava objekti samassa tilassa CustomerConnectin tilaserverissä, ainakin IP-puheluiden suhteen. Näitä kuvia pidetään yllä tilaserverin ja vaihdeserverin välillä kulkevien viestien avulla. Periaatteessa ylläolevan kuvan objekteihin tilaserveri ei koske mitenkään, vaikka se onkin samassa muistiavaruudessa niiden kanssa. Ainoastaan tilojenvälisiä yhtenevyystarkistuksia voidaan tehdä perustuen CSTA-mallien kuvien suoraan vertailuun. Luonnollisestikaan tilakuvat eivät ole täysin reaaliaikaisesti yhteneviä, koska muutokset IPX-serverin mallissa kuvaavat todellista tilannetta MCUssa, ja jos muutoksia tapahtuu esimerkiksi jonkin objektin tilassa, muutos saapuu ensiksi IPX:ään, ja vasta sitten tilaserveriin.

Globaalit muuttujat ovat samat kuin tilaserverin käytössä olevat (vaikkakin vain pointtereita itse objekteihin). CSwitchX esittää vaihdekomponenttia, ja on pääluokka muiden objektien käsittelemistä varten.

Siinä missä tilaserveri tekee päätökset mm. puheluiden reitityksistä, IPX:n tehtävänä on kommunikoida tilaserverin päätökset oikealle MCU:lle, ja MCU:n vastaus takaisin tilaserverille. IPXien tilaserveriltä saamat viestit perustuvat CSTA-mallin kuvan pohjalta muodostetuille päättelyille. Koko CSTA-mallia ei tässä käytetä, koska IP-puhelujen ja käytettyjen rajapintojen luonne ei taivu kaikkiin CSTAn vaatimuksiin. IPX muuntaa MCUlta saapuneet viestit CSTAn logiikan mukaisiksi. Joidenkin viestien tapauksessa tämä on melko suoraviivaista, mutta eräissä tapauksissa IPX:n on tutkittava CSTA-objekteja hyvinkin tarkasti.

Tällä hetkellä voi olla olemassa yhtäaikaa kaksi eri vaihdeserveriä: yksi PSTN-vaihdeyhteyttä varten ja IPX. PSTN-vaihde ja MCU-yhteyksiä varten olevien vaihdeserverien lähdekoodit käännetään toistaiseksi samaan ohjelmakirjastoon, cbxlib32.lib:iin. Tarkoitus on myöhemmin eriyttää koodi siten, että IPX-serveri olisi mahdollista kääntää dynaamisesti linkitettävään kirjastoon, IPXServer.dll:ään. Tällöin alustustiedoissa olevan asetuksen avulla voidaan määritellä, tuleeko IPX:n koodisivut ladata mukaan. (alustustiedot ovat tällä hetkellä CCServer.ini-nimisessä tiedostossa NT:n systeemihakemistossa.).

IPX-serveri käyttää kuvassa A.10 olevia luokkia, mutta vain abstrakteina kantaluokkina, joista se johtaa varsinaiset toteutukset. Kantaluokat sisältävät suurimman osan listojen käsittelyyn tarkoitetuista rutiineista, uusien objektien instantioinnin ja tarvittavat tietokantafunktiot. IPX:lläkin on yhteys tietokantaan, vaikka sitä ei ylläolevissa kuvissa esitetäkään, mutta yhteys on yksisuuntainen: IPX lukee tietokannasta laitteet, monitoroidut numerot (CustomerConnectin alueen sisällä olevien puhelinlaitteiden osoitteet) ja ryhmien tiedot (IPCC:llä kytkeytyvää asiakasta varten), joita se tarvitsee pystyttäessään CSTA-mallin kuvaa tilaserverin kuvaa vastaavaksi.

IPX:llä on abstraktit kantaluokat puhelu- yhteys- laite- ja vaihde-objekteja varten. Eri vaihdetyyppien erikoisoiminaisuudet on sijoitettu näistä luokista johdettuihin luokkiin. Esimerkiksi CCallX on abstrakti puhelua kuvaava kantaluokka. Se sisältää vain geneeristä koodia. IPX-serverin vaihdekohtainen implementointi tälle luokalle, CCallXIPX, sisältää mm. sisääntulevan IP-puhelun käsittelyrutiinit. Kun IPX-serveri luo puheluobjektista uuden instanssin, se luo CCallXIPX-objektin CCallX:n puhtaan virtuaalisen konstruktorin pakottamana. Tämän mukana tulee periyttämisen kautta joitakin geneerisiä puhelunkäsittelyrutiineja itse CCallX-kantaluokasta.

1.3.5 Asennus

IPX-serverin asentaminen ei vaadi muuta kuin tiettyjen parametrien asentamista serverin ja MCUn alustustietoihin, sekä serverin kääntämistä IPX_SRV-direktiivin kanssa, jonka jälkeen tarvitut funktiot ovat osa serverin ajotiedostoa.

1.3.6 Tiedostot

Projekti tehtiin Microsoftin ympäristössä: käytetty kääntäjä oli Microsoft C++ 4.2:n mukana oleva kääntäjä ja käyttöjärjetelmänä Windows NT Server 4.0. Projektiin sisältyvät tiedostot ovat:

- IPXServer.mdp (työtila- ja projektitiedosto)
- IPXServer.cpp (kirjaston päätiedosto)
- IPXswi.cpp (Vaihdeluokan lähdekoodi)
- IPXcal.cpp (CSTA-puheluluokan lähdekoodi)
- IPXcon.cpp (CSTA-yhteysluokan lähdekoodi)
- IPXdev.cpp (CSTA-laiteluokan lähdekoodi)
- IPXmsg.cpp (eri rajapintojen viestien käsittelyyn liittyvä lähdekoodi)

2 Muutokset CustomerConnectin tilapalvelimeen

2.1 Useiden vaihdepalvelimien käsittely

CustomerConnectin version 1.5 tilapalvelin oletti vaihdepalvelimia ja vaihdekomponentteja olevan ainoastaan yhden (VRU-yksikkö toimii erikoislaatuisena clienttina oman rajapintansa kautta), joista vaihdekomponentin piti olla käynnistettynä, ennenkuin tilaserveri pystyi käynnistymään. Erityyppisiä vaihdekomponentteja tuettiin, mutta näistä saattoi vain yksi kerrallaan olla käytössä. Suurin muutos olikin tämän oletuksen poistaminen, ja useiden yhtäaikaisten vaihteiden vaatiman toiminnallisuuden ja tunnistusmekanismien lisääminen olemassaolevaan koodiin. Toistaiseksi vaihdekomponenttien *välillä* ei oleteta olevan minkäänlaista vuorovaikutusta, mikä yksinkertaistaa puhelunkäsittelyä huomattavasti tarvituissa muutoksissa laskien. (Niin kauan kuin järjestelmässä ei ole IP-PSTN-yhdyskäytäviä, tämä oletus on täysin oikeutettu)

Tehdyt muutokset on listattu alla:

- Globaalit muuttujat muutettiin globaaleiksi, entistä globaalin muuttujan luokan instansseja sisältäviksi listoiksi.
- Vaihdeserverin viestijonoa muutettiin siten, että se kykenee kuuntelemaan verkkoa
- Tietoliikennekirjastoihin lisättiin mahdollisuus luoda viestijonoja dynaamisesti
- CMsg::Send voi käyttää viestin jäsenmuuttujana olevaa viestijonoa
- uusia vaihteita pystyy lisäämään dynamisesti (testaus kesken)
- vaihdekomponentin ei tarvitse olla läsnä kaikkien vaihdeserveri-tyyppien käynnistyksen yhteydessä
- vaihdeserverit tunnistetaan CSTA-yhteysobjektin tunnisteesta. Tunniste on kokonaisluku, joka muodostetaan itse yhteyden tunnisteesta ja vaihteen tunnisteesta seuraavasti CID_G=10000*SWID+ CID_L. Tässä CID_G on globaali yhteystunniste ja CID_L lokaali, vaihdekohtainen yhteystunniste. SWID tarkoittaa vaihteen tunnistetta. Vastaavasti SWID =⌊CID_G/10000⌋, ja CID_L=CID_G (mod 10000), missä ⌊x⌋ tarkoittaa lattiafunktiota, eli tulosta lähinnä pienempää kokonaislukua.

Ensimmäinen mutos – globaalien muuttujien muuttaminen globaaleiksi listoiksi – johtui juuri edellä mainitusta yhden vaihdekomponentin oletuksesta. Jos systeemissä on yhtäaikaa useampia vaihdekomponentteja käytössä, niitä tulee käsitellä listojen eikä staattisten globaalimuuttujien kautta. Tämä toi mukanaan tunnistusmekanismin tarpeellisuuden: X-rajapintaa käyttävät useat vaihdeserverit, joten täytyi löytää keino erotella TAPI- IP- ja Siemens-vaihdekomponenttia koskevat viestit toisistaan. Loppujen lopuksi

viestejä käsitellään samalla tavalla, mutta viestin mukana aina kulkeva CSTA-yhteys-objektin tunniste määrää viestin tarkoittaman vaihteen, ja ero tehdään CSTA-objektien eri toteutuksissa.

Koska IPX-serverin takana oleva vaihdekomponentti on IP-verkossa liikennöivä prosessi, ei useammalle viestijonolle samassa loogisessa avaruudessa tule tarvetta, mikäli yhteinen viestijono pystyy kuuntelemaan verkkoa. Tämä osoittautui sikäli ongelmalliseksi, että tietoliikennekirjasto (COMLIB) salli aluksi vain yhden verkkoa kuuntelemaan pystyvän viestijonon. COMLIBiin tehtyjen muutosten jälkeen vastaan-ottava viestijono ei ole enää kiinnitetty `G_pServerQueue`:ksi, vaan on `CMsg`-luokan jäsenmuuttujana.

Aiemmin vaihdekomponentti piti käynnistää ennen kuin tilaserveri pystyi kunnolla alustumaan. Tämä tarkastettiin `CSwitch::Connect()`-metodin avulla. Kun kaikki laitteet ovat dynaamisia luonteeltaan, ja alkutilanteessa vaihdetta ei välttämättä ole paikalla, vaikka kysymyksessä ei olekaan virhetilanne, ei Connect()-metodi saa palauttaa virhetilannetta, vaikka vaihdekomponentti puuttuukin.

3 Laitteistokomponentit

Olettaen, että CustomerConnectin itsensä tarvitsema laiteympäristö on jo olemassa, ei IP-yhteyden mukaantuominen aseta suuria vaatimuksia laitteistolle. TCP/IP-protokollalla toimiva verkko on ehdoton vaatimus, samoin kuin riittävät siirtonopeudet ja prosessointinopeus yksikössä. Eri medioihin tarvittava laitteisto ei ole pakollinen, mikäli kyseistä mediaa ei haluta käyttöön: itse CustomerConnectin IP-yhteys ei ota kantaa käytettyyn mediaan, se voi olla yhtä hyvin tesktiä kuin audiotakin tai jopa täysimittainen videokonferenssi.

Jos pelkkä chat-viestien lähettäminen on riittävää, 9600 bps modeemi riittää kyllä tähän. Mutta mitä enemmän, monipuolisempaa ja laadukkaampaa mediavirtaa verkossa siirretään, sitä edistyneempi laitteisto sen käsittelemiseen vaaditaan. Taulukko A.7 selittää vaaditut nopeudet ja muun laitteiston. Taulukko A.7 ottaa kantaa ainoastaan IP-puheluohjelmaa käyttävälle tietokoneelle

Taulukko A.7: Laitteistovaatimukset IP-puheluohjelmalle

Media	Yhteysnopeus	Tarvittu RAM	Prosessor in nopeus	Muu laitteisto
Chat / Talk	2400 bps (modeemi)	8 MB	> 66 MHz	-
Audion vastaanottaminen	> 14,4 kbps (modeemi, ISDN tai nopeampi)	8 MB (W95) 16MB (NT)	> 66 MHz	Äänikortti ja kaiuttimet

Audio vastaanottaminen & lähettäminen	> 14.4 kbps (modeemi, ISDN tai nopeampi)	8 MB (W95) 16MB(NT)	> 66 MHz	Kuten yllä + mikrofoni
Videon vastaanottaminen	> 28.8 kbps (modeemi, ISDN tai jokin T1-yhteys)	16 MB (W95) 24 MB (NT)	> 90 MHz	Kuten yllä + Videokortti yhteyden nopeuttami-seksi.
Videon vastaanottaminen & lähettäminen	> 28.8 kbps (modeemi, ISDN tai T1-yhteys.)	16 MB (W95) 24 MB (NT)	> 90 MHz	Kuten yllä + videokamera

4 Toiminnalliset kuvaukset

4.1 Käynnistäminen

Jokaisen CustomerConnectin IP-toiminnallisuuteen kuuluvan ohjelmistokomponentin asennus ja käynnistys parametreineen on kuvattu erikseen jokaisen komponentin yhtyedessä, tai itse komponentista kertovassa yleiskuvauksessa. Tässä esitetään koko järjestelmää koskeva käynnistyssekvenssi.

Ensimmäisenä käynnistetään CustomerConnect tilaserveri, joka alustaa sekä PSTN- että IPX-vaihdeserverit. IPX alkaa käynnistyksensä jälkeen kuunnella verkkoa mahdollisten MCU-komponenttien yhteydenottojen varalta. Kun jossakin päin verkkoa sitten käynnistetään MCU (mieluummin samassa LAN:ssa, muttei välttämättä yrityksen palomuurin sisäpuolella), se lukee ensin levyltä alustustietonsa, ja kytkeytyy niiden mukaisesti sopivaan IPX-serveriin. Kytkeytyminen sisältää autentikoinnin väärien MCU:iden kytkeytymisen estämiseksi.

Jos MCU ei löydä IPX-serveriä heti, se jatkaa tämän etsimistä, kunnes jokin IPX joko löytyy tai MCU suljetaan. MCU:n IPX:ään kytkeytyminen katsotaan alustusvaiheeksi, jolloin MCU ei hyväksy yhtäkään IPCC:tä, joten yksikään asiakas ei pysty ottamaan yhteyttä järjestelmän palveluihin tänä aikana. (*Huom*: MCU:n on aloitettava verkon kuunteleminen ennen kuin se voi kytkeytyä IPX:ään, koska IPX:stä päin synkronista kanavaa ei saada luotua, jos MCU:n viestijono ei ole kuuntelevassa tilassa. Periaatteessa siis IPCC:lle on toistaiseksi olemassa sellainen hetki, milloin se pystyy kytkeytymään järjestelmään, vaikka gatekeeperiä ei vielä ole löydetty. Tämä tapahtuu äärimmäisen harvoin, ja tuloksena on vain IPCC, joka ei saa minkäänlaisia vastauksia pyyntöihinsä.)

Jos MCU:n onnistui kirjautua IPX-serveriin, ja toisaalta yksikään IPCC ei pyytänyt palvelua ennen tätä, MCU:n sisäinen tila muuttuu normaaliksi, ja se voi aloittaa viestien vastaanottoprosessin: IPCC:den kirjautumiset ja palvelupyynnöt samoin kuin IPX:n komennotkin.

Kun asiakas haluaa lähettää palvelupyynnön, hän lataa ensin jonkin ennaltamääritellyn web-sivun yrityksen palvelimelta. Kun sivu on latautunut ensimmäisen kerran, selain käynnistää siihen upotetun java-appletin. Appletin käynnistysvaiheessa se ottaa yhteyttä parametrilistan mukaiseen MCU:n osoitteeseen. Jos MCU:ta ei pystytä tavoittamaan, applet tulostaa virheilmoituksen ja tarjoaa asiakkaalle jonkin vaihtoehtoisen kommunikointikanavan, esim. mahdollisuuden lähettää sähköpostia. Appletin ollessa pelkkä yksittäinen nappi tällaista ei tapahdu, koska MCU:hun ei oteta yhteyttä ennen napin on painamista.

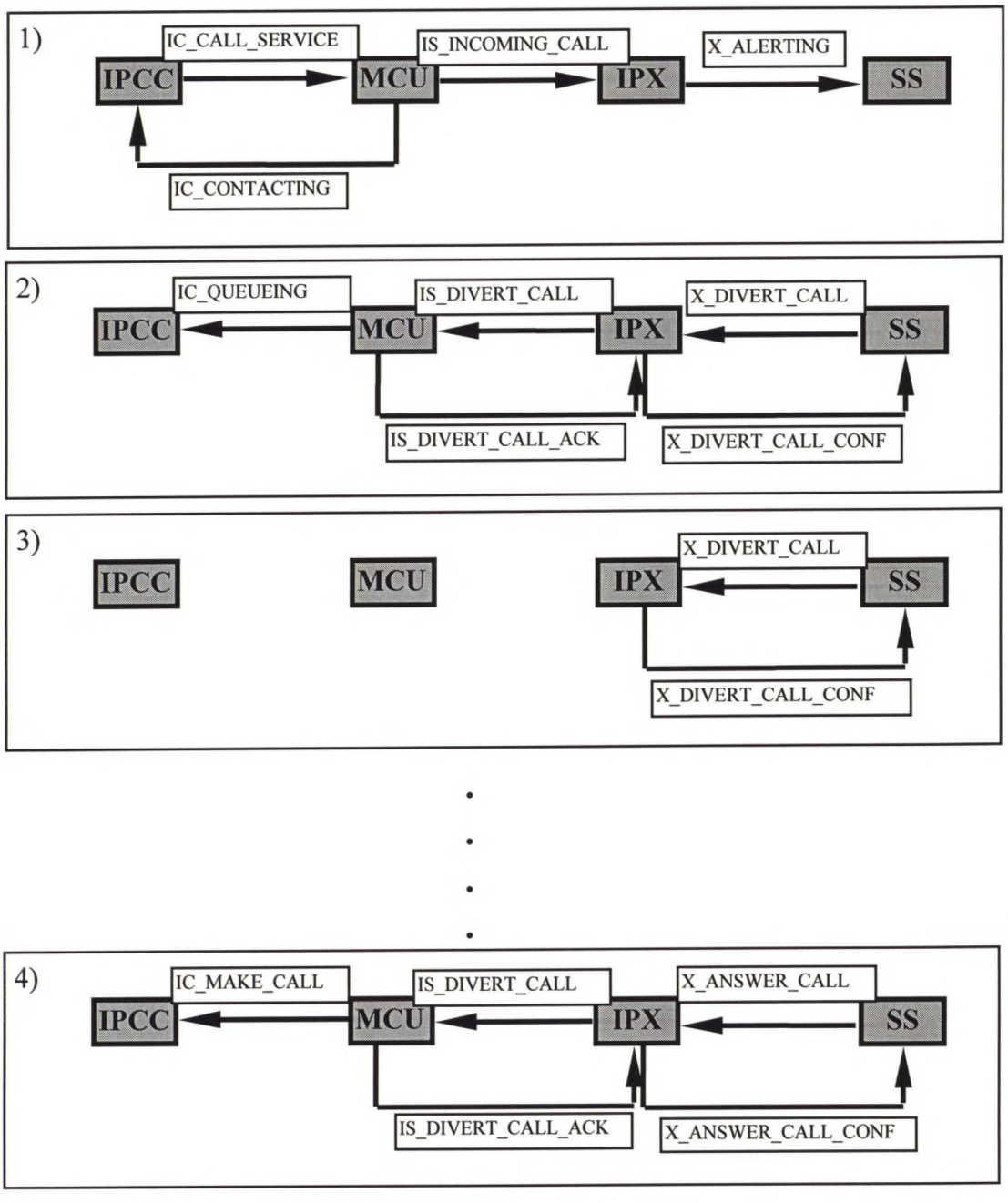
Jos applet löytää MCU:n ja kykenee tunnistautumaan hyväksyttävästi järjestelmälle, applet alustaa käyttöliittymänsä, ja lähettää MCU:lle viestin CustomerConnectin tietokannassa olevien ryhmä/palvelutietojen hakemisesta. Jos MCU on ylhäällä normaalissa tilassa, sillä on jo hallussaan kaikki ryhmätiedot, sillä ne haetaan kannasta valmiiksi jo käynnistysvaiheessa, heti kun yhteys CustomerConnect-serveriin on muodostettu. Tässä ei siis tapahdu ylimääräistä prosessointia jokaisen IPCC:n kytkeytymisvaiheessa, vaan prosessointitaakka päättyy MCU:hun IPCC:ltä CustomerConnect-clientiin saakka johtavassa ketjussa. Jos MCU:ta pidetään pidemmän aikaa pystyssä, ryhmädataa on aika ajoin virkistettävä. Tämä tehdään erilaisten ajastettujen viestitapahtumien avulla.

Ryhmädata (palveluiden nimet) näytetään appletissa dynaamisena listana tai nappijoukkona riippuen saatavilla olevien ryhmien määrästä, ja asiakas voi painaa joko suoraan ryhmän nimellä tunnistettavaa nappia, tai valita listasta ryhmän, ja painaa ”Contact”-nappia erikseen.

CustomerConnectin valvonnassa olevien agenttien on aukaistava tai asetettava automaattisesti aukaistaviksi IP-puheluohjelmansa autoanswer-tilaan aina NT:n loginin yhteydessä, jotta CustomerConnect pystyisi ohjaamaan IP-puheluita näihin oikein.

4.2 Yhteydenotto palveluun (ryhmäpuhelu)

Yhteydenotto palveluun vaatii ylläkerrotun kaltaiset toimenpiteet, ts. sen, että asiakkaalla on oltava yrityksen yhteydenottoa varten tarjoama web-sivu auki Microsoftin turvallisuuskäytäntöä tukevassa selaimessa. Tämän jälkeen, kun asiakas on käyttöliittymän kautta ilmaissut haluavansa ottaa yhteyttä johonkin palveluun, ja olettaen, että vaaditut komponentit ovat kaikki toiminnassa, seuraavanlainen tapahtumasarja käynnistyy:



Kuva A.11: Yhteydenotto palveluun: viestitapahtumat

Kuten nähdään kuvan A.11 kohdasta 1, välittömästi ”contact”-napin painamisen jälkeen IPCC lähettää **IC_CALL_SERVICE**-viestin MCU:lle. MCU vastaa tähän heti **IC_CONTACTING**-viestillä, vain informoidakseen, että se on vastaanottanut viestin. (IPCC:tä ei voi jättää synkronisesti odottamaan mahdollista agentin vastausta, koska ei ole mitään takeita siitä, että asiakas ylipäänsä pysyy juuri ladatulla sivulla.) MCU:n vastattua IPCC:lle on seuraava toimenpide lähettää **IS_INCOMING_CALL** IPX-serverille. IPX tekee omat käännöksensä viestistä, luo asianomaiset CSTA-rakenteet ja –tunnisteet, ja lähettää lopuksi **X_ALERTING**-viestin tilaserverille.

Tilaserveri tekee omat päättelynsä sisääntulevasta puhelusta, ja käskee sen ensin siirrettäväksi kutsunsiirrolla pois ryhmän loogisesta numerosta joko hälyttämään

jonopaikkaan, tai vastattuna IVR-systeemille. Tilaserveri lähettää IPX:lle **X_DIVERT_CALL**in vapauttaakseen ryhmän loogisen numeron. Koska tämä ei alunperinkään ollut mitenkään varattu resurssi, IPX antaa **IS_DIVERT_CALL**in avulla MCUn vain tietää, että tilaserveri käsittelee jo palvelupyyntöä. MCU vuorostaan kertoo tämän takaisin IPCC:lle **IC_QUEUEING**-viestillä, ja vahvistaa IPCC:n olevan edelleen saavutettavissa lähettämällä **IS_DIVERT_CALL_ACK**in takaisin IPX:lle. Kun IPX saa vahvistusviestinsä kutsunsiirtoon, se lähettää vuorostaan **X_DIVERT_CALL_CONF**in tilaserverille (kuva A.11, osa 2).

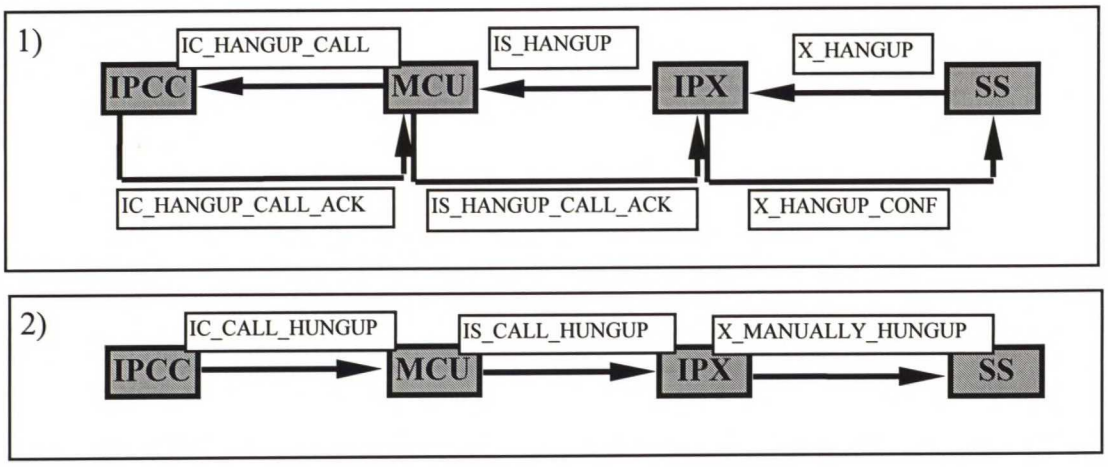
Kuvan A.11 3-osan esittämä vaihe näyttää, kuinka vaihdeserveri saattaa divertoida puhelua useamman kerran (esim. henkilökohtaisten reititysten yhteydessä) eri numeroiden ja agenttien välillä, kunnes puheluun vastaamaan halukas vapaa agentti löytyy. Tällä kerralla tilaserveri ensin käskee kutsunsiirron agentin loogisesta numerosta tämän alaliittymään, ja vasta puhelun ollessa alaliittymässä (päätteellä) hälyttämässä tehdään seuraavan kerran jotakin IPCC:n kannalta merkityksellistä. Kaikkiin näihin IP-puhelun kannalta turhiin kutsunsiirtoviesteihin (myös siirto agentin loogisesta numerosta tämän varsinaiseen numeroon) IPX reagoi vain vahvistamalla sen **X_DIVERT_CALL_CONF**-viestillä lähettämättä mitään viestejä MCU:lle, koska varsinaista mediavirtayhteyttä ei vielä ole synnytetty.

Vasta kun tilaserveri lähettää **X_ANSWER_CALL**in, IPX lähettää jotakin MCU:lle (**IS_DIVERT_CALL**in), jolloin tämä ilmoittaa IPCC:lle **IC_MAKE_CALL**illa puhelun aloittamiskäskystä. IPCC vahvistaa kutsunsiirron **IS_DIVERT_CALL_ACK**illa, jonka perusteella IPX voi lähettää **X_ANSWER_CALL_CONF**in vastaukseksi kutsunsiirtopyyntöön.

Nyt IP-puheluohjelma avaa yhteyden, ja – olettaen, että agentilla on IP-puheluohjelmansa autoanswer-tilassa – aloittaa mediavirran lähettämisen.

4.3 Yhteyden katkaisu

Puhelun katkaisussa on oikeastaan kolme eri tapausta huomioitavana. Katkaisukomento voi tulla tilaserveriltä (jos agentti katkaisee puhelun CustomerConnect-clientin puheluikkunasta käsin, tai jos tilaserveri itse päättää lopettaa puhelun), IPCC:ltä (asiakas katkaisee puhelun IPCC:n käyttöliittymästä) tai IP-puheluohjelmalta (agentti tai asiakas käyttää minimoitua IP-puheluohjelman ikkunaa katkaisuun, tai IP-puheluohjelma itse katkaisee puhelun menetettyään esim. konferenssi-isäntänsä). Kolmas tapaus redusoituu toiseen, koska IP-puheluohjelman rajapinnalle lähettämät viestit paljastavat myös IPCC:n kautta annetun katkaisukomennon. Ensimmäiseen tapaukseen liittyvä viestiliikenne on esitetty kuvan A.12 ensimmäisessä osassa, ja vastaavasti toiseen tapauksen liittyvä liikenne kuvan toisessa osassa.



Kuva A.12: viestiliikenne puhelun katkaisun yhteydessä

Jos tilaserveri määrää katkaisun, ei ole olemassa takeita sen onnistumisesta. Siksi kutakin pyyntöä vastaava vahvistusviestien sarja lähetetään silloin, kun edellinen taso on saanut päätöksen toiminnan onnistumisesta tai epäonnistumisesta. Viestien lähetysjärjestys kiertää vastapäivään lähtien tilaserveristä (SS).

Jos puhelu on katkaistu ns. manuaalisesti, mitään ei voida tehdä enää sen estämiseksi, koska mediavirta sulkeutuu sillä hetkellä, kun asiakas painaa katkaisunappia. Tämä ei tietenkään päde IPCC:n kontrollien kohdalla, mutta tämän liittymän valvominen ja katkaisun luvan hakeminen tilaserveriltä ennen varsinaista katkaisua ei juuri kannata: jos asiakas ei huomaa IPCC:stä annetun käskyn tuottavan tulosta, hän voi ottaa esille IP-ohjelman, ja tehdä katkaisun sieltä käsin. Puhelu siis katsotaan jo menetetyksi, kun tieto siitä saapuu MCU:lle saakka, ja tästä informoiva viesti lähetetään tilaserverin tasojen suuntaan. Vahvistusviestejä tälle ei lähetetä millään tasolla.

Kummassakin tapauksessa, saavat IPX ja MCU sitten manually-hungup-viestin tai vahvistuksen onnistuneesti katkaistuun puheluun, molemmat siivoavat CSTA-rakenteitaan (MCU riisuttua CSTA-malliaan) ja vapauttavat puhelun käyttämän muistin.

5 Palomuuri ja IP-puhelin

CustomerConnectin todennäköisin sijoitus tulee olemaan yrityksen intranetissä palomuurin suojissa. Palomuurin on täytettävä tiettyjä vaatimuksia, tai se pitää pystyä vähintään konfiguroimaan sellaiseksi, ennenkuin CustomerConnectin IP-puoli voidaan ottaa käyttöön. Itse IP-puheluohjelma asettaa suurimman osan näistä vaatimuksista: tiettyjen porttien tulee olla auki ainakin ulospäin suuntautuvalle liikenteelle, sekä joitakin dynaamisia portteja vaaditaan TCP:tä käyttäville kontrolliviesteille ja mediavirran käyttämälle RTP:lle UDP:n päällä. Reititin-palomuurien kanssa tämän ei pitäisi olla ongelma, mutta proxy-tyyppiset muurit eivät yleensä tue UDP:n käsittelyä.

Ongelmia voi syntyä erityisesti silloin, kun yhteys avataan ulkoa käsin (kuten on tarkoitus sisääntulevassa IP-puhelussa) Normaalisti satunnaiset yhteyspyynnöt ulko-

puolelta tulisi estää. Melkein kaikki palomuuriohjelmistot sallivat jonkinasteista sisääntulevan liikenteen uudelleenohjausta, mutta saattaa olla, että joissakin tapauksissa on tyydyttävä puhelun aloittamiseen palomuurin sisäpuolelta.

Tunnetuista palomuuriohjelmistoista ainakin FireWall-1 sopii UDP:n päällä toimivien kommunikointisovellusten palomuuriksi. Seuraava lainaus on Microsoftin Knowledge Base:sta:

"*Lisäksi, jotkin palomuurit pystyvät päästämään TCP:tä lävitseen tietyistä porteista, ja sekundäärisiä UDP-yhteyksiä dynaamisesti asetetuista porteista, mutta* ***eivät pysty tarjoamaan osoitteenkäännöstä mielivaltaiselle määrälle sisäisiä IP-osoitteita, tai eivät pysty tekemään niin dynaamisesti****. Tämäntyyppisten palomuurien kanssa voidaan synnyttää IP-puheluyhteyksiä (audio- ja videovirralla) yrityksen palomuurin sisältä ulkomaailmaan, mutta* ***ei pystytä aikaansaamaan yhteyksiä palomuurin ulkopuolella olevista tietokoneista palomuurin sisällä oleviin tietokoneisiin****.* "

Yhteenvetona: palomuurille asetetut vaatimukset ovat: (ensimmäisessä versiossa)

NetMeetingin virheettömän toiminnan asettamat vaatimukset:

- Seuraavat portit aukaistava liikenteelle:
  - 389 ILS:lle TCP:n päällä
  - 522 ULS:lle TCP:n päällä
  - 1503 T.120 TCP:n päällä
  - 1720 H.323-puhelunalustukselle TCP:n päällä
  - 1731 audio-puhelukontrollille TCP:n päällä
  - 1025..65536 (dyn.) H.323 puhelukontrollille TCP:n päällä
  - 1025..65536 (dyn.) mediavirroille RTP:nä UDP:n päällä
- Seuraavat protokollat päästettävä lävitse
  - TCP (primäärinen)
  - RTP UDP:n päällä (sekundäärinen)
- Sisäiset IP-osoitteet on pystyttävä kuvaamaan ulospäin (sisääntulevien puheluiden kunnollista toimivuutta varten)

IP-vaihteen virheettömän toiminnan asettamat vaatimukset:

- Kolme porttia IPX-serveri – MCU-yhteyttä varten (kolme TCP/IP-sockettia)

- Sisäiset IP-osoitteet on pystyttävä kuvaamaan ulospäin (järjestelmä pakottaa yhteyttä haluavan oapuolen aloittamaan puhelun)

Liite B:

CustomerConnect 2.0 IP-yhteyden viestien vastaavuus H.323:n RAS- ja puhelusignalointikanavien viestien kanssa

RAS-kanavan viestit

Viesti	Nimi	GK ->T	T-> GK	Tarkoitus	CC-vastaavuus	Huomioitavaa
ACF	Admissions Confirmation	X		Kaistanleveys varattu ARQ:n lähettäneelle päätepisteelle. Puhelusignalointi voi alkaa.	Ei tukea	
ARJ	Admissions Reject	X		Kaistanlevelyttä ei saatu päätepisteen mediavirralle.	Ei tukea	Tämä on vastauksena myös, jos soitetun osapuolen GK on valinnut reititetyn signaloinnin. Mukana on tällöin parametri: *cause=**routeCallToGatekeeper**.*
ARQ	Admissions Request		X	Alustava pyyntö varata tietty kaistanleveys päätepisteen käyttämälle mediavirtojen tarvitsemalle datalle poislukien protokolla-overhead.	Ei tukea	ARQ:lla kysytään GK:lta lupa mediavirran käytölle. Ennen luvan saamista ei aloiteta edes puhelusignalointia.
BCF	Bandwidth Confirmation	X		GK hyväksyy kaistanlevelyden muutoksen.	Ei tukea	
BRJ	Bandwidth Reject	X		GK ei salli kaistanleveyden muutosta (joko rekisteröitymätön päätepiste tai resurssipula)	Ei tukea	
BRQ	Bandwidth Request		X	Päätepiste haluaa kasvattaa sille varattua kaistanleveyttä.	Ei tukea	Voidaan käyttää myös kaistanleveyttä pienennettäessä, mutta tätä suositellaan ainoastaan pitempiaikaisessa kaistanlevelyden pudotuksessa.
DCF	Disengage Confirmation	X	X	Päätepiste vastaa tällä aina GK:n DRQ:iin, ja GK päätepisteen DRQ:iin, jos kaistanleveys pystytään vapauttamaan.	**IC_HANGUP CALL_ ACK**	Positiivinen paluuarvo. Tämä kulkee vain päätepisteeltä GK:lle menevän DCF:n mukana. GK ei vastaa DCF:llä päätepisteen DRQ:hun, koska sillä ei ole mitään valtaa tähän aspektiin.

DRJ	Disengage Reject	X		Jos kaistanlevelyttä ei pystytä vapauttamaan (rekisteröitymätön päätepiste, olematon kaistanleveys, tai kilpailutilanne)	**IC_HANGUP CALL_ ACK**	Negatiivinen paluuarvo. Tämä kulkee vain päätepisteeltä GK:lle menevän DCF:n mukana. GK ei vastaa DCF:llä päätepisteen DRQ:hun, koska sillä ei ole mitään valtaa tähän aspektiin.
DRQ	Disengage Request	X	X	Päätepiste ei tarvitse enää kaistanleveyttä (joko mediavirta loppuu, tai GK:n resurssien vähyys pakottaa poistamaan päätepisteelle varatun kaistanleveyden.)	**IC_HANGUP CALL, IC_CALL_H UNGUP**	**IC_HANGUP_CALL** GK:lta IPCC:lle; ja **IC_CALL_ HUNGUP** IPCC:ltä GK:lle.
GCF	Gatekeeper Confirmati on	X		GK:n vastaus GRQ:iin: päätepisteen sallitaan rekisteröityä. (ei vars. rek. viesti)	Ei vars. tukea. Jos yhteydenluonti MC:hen onnistuu, katsotaan tämä yhteneväksi GCF:n kanssa.	
GRJ	Gatekeeper Reject	X		Jos GRQ on lähetetty suoraan tietylle GK:lle broadcastin sijaan, GK voi tämän avulla kieltäytyä rekisteröimästä päätepistettä.	Ei vars. tukea. Jos yhteydenluonti MC:hen ei onnistu, katsotaan tämä yhteneväksi GRJ:n kanssa.	
GRQ	Gatekeeper Request		X	Gatekeeperin etsiminen jostakin GK-pilvestä.	Ei vars. tukea. Esiintyy alustusdatassa.	
IRQ	Information Request	X		GK lähettää päätepisteelle kyselyn tämän toiminnan statuksesta. (**ei** puhelun status)	**Z_TIMEOUT** (*)	(*) Tietoliikennekerroksen tukema viesti **Z_TIMEOUT** ilmestyy molemmille kommunikoiville osapuolille silloin, kun tietoliikennekana-vassa ei ole ollut liikennettä tiettyyn aikaan.
IRR	Information Request Response		X	Päätepisteen vastaus IRQ:iin toiminnan statuksesta. Voidaan myös lähettää ilman IRQ:takin	**Z_TIMEOUT** (*)	

				eräänlaisena ”sydämenlyöntinä ”.		
LCF	Location Confirma-tion	X		GK:n vastaus aliaskyselyyn, jos osoite löytyi. Osoite parametrinä.	**IC_MAKE-CALL**	LCF ja vaadittu osoite ovat osana tätä viestiä, joka kylläkin sisältää myös H.225.0:n konferenssin luonti-viestien toiminnallisuutta.
LRJ	Location Reject	X		Niiden GK:den vastaukset, joista aliaksen kuvaamaa osoitetta ei löytynyt.	**IC_ERROR**	Syynä joko GK:n tavoittamattomuusm tai tyhjä ryhmä (ei agentteja)
LRQ	Location Request		X	Päätepisteen kysely jonkin aliaksen osoitteesta.	**IC_CALL-SERVICE**	Parametrina ryhmän ”alias”, eli nimi tai ID.
RCF	Registratio n Confirm	X		GK hyväksyy rekisteröitymisen (myös duplikaatti-rekisteröinnin, jos aliakset ovat erinimisiä)	**Z_CLIENT-CONNECT-ED, IC_LOGIN-ACK** (ei toteutettu)	ks. RRQ
RRJ	Registratio n Reject	X		GK hylkää rekisteröitymisen. Se voi tehdä näin omista syistään, mutta ainakin silloin, jos aliasten ja osoitteen välillä on epäselvyyksiä.	**IC_ERROR**	Saadaan, jos GK ei ole valmiustilassa. ks. RRQ
RRQ	Registratio n Request		X	Päätepiste rekisteröityy GK:lle, l. ilmoittaa sille osoitteensa aliaksineen.	**IC_LOGIN, Z_CLIENT_ CONNECTE D**	RRQ on toistaiseksi osana itse yhteydenottoa: GRQ ja RRQ kulkevat samassa yhteydenluonnissa. IC_LOGIN on tarkoitettu RRQ:n implementoinniksi.
UCF	Unregister Confirmati on	X	X	Päätepiste poistettiin GK:n hallinnasta. T vastaa aina UCF:llä GK:n lähettämään URQ:hun.	Ei tukea toistaiseksi, koska asiakas voi milloin tahansa sulkea päätepisteen.	
URJ	Unregister Reject	X		Päätepisteen poistoa GK:n hallinnasta ei hyväksytty. Tähän	Ei tukea toistaiseksi, koska asiakas voi milloin	IPX saa tiedon monitoroinnin lopettamisesta kyseisen IPCC:n kohdalta

				voi olla syynä GK:n sisäinen tila, mutta useimmiten olematttoman päätepisteen poistoyritys.	tahansa sulkea päätepisteen.	**X_MONITOR_DEVICE-**viestin parametrinä
URQ	Unregister Request	X	X	Päätepiste pudottautuu (tai pudotetaan) pois GK:n hallinnasta.	**Z_CLIENT_ DIS-CONNECTED, IC_ERROR**	**Z_CLIENT_DISCONNECTED** kulkee IPCC:ltä MC:lle ja **IC_ERROR** toiseen suuntaan. **IC_ERROR** kertoo MC:n takana olevan GK:n tai itse MC:n kieltäytyvän tarjoamasta palveluja

Puhelusignalointikanavan viestit (H.225.0)

Tässä taulukossa käsitteellä "konferenssi" viitataan myös kahdenväliseen ts. vain kaksi jäsentä sisältävään konferenssiin, joka normaalissa kielenkäytössä ymmärrettäisiin puheluksi. Tässä sen sijaan puhelut ovat konferenssiobjektissa kiinni olevia objekteja, mikäli jokin puheluobjektiin liityvä päätepiste osallistuu konferenssiin.

Viesti	GK ->T	T-> GK	Parametrit	Tarkoitus	CC-vastaavuus	Huomioitavaa
Setup	X	X	CRV, Kohteen osoite, CID, viestin tarkoitus (esim. create, invite, join)	Päätepiste alustaa konferenssin, tai muuttaa sen kokoonpanoa.	**IC_CALL-SERVICE**	Voi kulkea myös GK:n kautta reititetyssä puhelusignaloinnis-sa. IC_CALL_SERVICE sisältää myös LRQ:n, joten H.225.0-viestejä on yhdistetty RAS-kanavan viesteihin.
Call Pro-ceeding	X	X	CRV	Statusviesti toiselle päätepisteelle: konferenssi käynnistyy, muttei vielä hälytä.	**IC_CON-TACTING**	Voi kulkea myös GK:n kautta reititetyssä puhelusignaloinnissa
Alert-ing	X	X	CRV (Call Reference Value)	Statusviesti toiselle päätepisteelle: konferenssi hälyttää.	**IC_QUEUE-ING**	Voi kulkea myös GK:n kautta reititetyssä puhelusignaloinnissa
Con-nect	X	X	CRV, CID (Conference ID), käynnistyvän H.245-kanavan TA (Transpoint Address)	Setup onnistunut, H.245-kanava voidaan aukaista.	**IC_MAKE-CALL**	Voi kulkea myös GK:n kautta reititetyssä puhelusignaloinnissa
Facility		X	CRV, CID, uuden reitittäjän osoite (TA), uuden reitittäjän tyyppitieto (esim. routeCallToMC, routeCallToGK)	Soitetun päätepisteen GK estää suoran H.245- ja puhelusignaloinni n. Soittajan käytettävä uutta, täsä viestissä tulevaa osoitetta.	Ei tukea	*Facility* lähetetään GK:lle vain siinä tapauksessa, että soitetun osapuolen GK on valinnut reititetyn puhelusignaloinnin, ensimmäisen *Setup*-viestin lähettäjänä on soittavan osapuolen GK. Muutoin *Facility* on aina päätepisteiden välinen.
Release Comp-lete	X	X	CRV, syy negatiiviseen paluuarvoon.	Mikäli jokin toiminta ei onnistu, sen varaamat resurssit vapautetaan, ja kerrotaan	**IC_ERROR, IC_HANGUP CALL_ACK**	*Release Complete*n lähettää GK vain siinä tapauksessa, että se on saanut *Facility*-viestin kerrotussa tilanteessa. **IC_ERROR** saadaan

				epäonnistuminen kutsuvalle osapuolelle.		GK:lta, kun jokin toiminto epäonnistuu ja **IC_HANGUP_CALL_ACK** IPCC:ltä, jos puhelunkatkaisu epäonnistuu/onnistuu. Muissa tapauksissa IPCC ei välitä Release Complete – viestejä GK:lle
Status-Enquiry	X		CRV	GK kyselee päätepisteeltä konferenssin tilaa.	Ei tukea	
Status		X	CRV, konfenssin tila.	Päätepiste lähettää GK:lle konferenssin tilan.	Ei tukea	Voidaan lähettää myös pyytämättä.
Close-Logical Channel	*)	*)	CRV, kanavan osoite	Jos BRQ menee lävitse, vanha looginen kanava on ensin suljettava.	Ei tukea	*) Käytetään BRQ:n yhteydessä
Open-Logical Channel	*)	*)	CRV, uuden kanavan osoite	Jos BRQ menee lävitse siirrytään uuteen, laajakaistaisempaan kanavaan.	Ei tukea	*) Käytetään BRQ:n yhteydessä
Open-Logical ChAck	*)	*)	CRV	Toinen päätepistekin on nyt siirtynyt uuteen kanavaan.	Ei tukea	Käytetään BRQ:n yhteydessä